AUTOMATED IRIS RECOGNITION SYSTEM USING CMOS CAMERA WITH PROXIMITY SENSOR

by

Paulo R. Flores
Hazel Ann T. Poligratis
Angelo S. Victa

A Design Report Submitted to the School of Electrical Engineering, Electronics Engineering, and Computer Engineering in Partial Fulfilment of the Requirements for the Degree

Bachelor of Science in Computer Engineering

Mapua Institute of Technology
September 2011

Approval Sheet

Mapua Institute of Technology School of EECE

This is to certify that we have supervised the preparation of and read the design report prepared by **Paulo R. Flores**, **Hazel Ann T. Poligratis**, and **Angelo S. Victa** entitled **Automated Iris Recognition System Using CMOS Camera with Proximity Sensor** and that the said report has been submitted for final examination by the Oral Examination Committee.

Engr. Ayra G. Panganiban
Design Adviser

As members of the Oral Examination Committee, we certify that we have examined this design report, presented before the committee on **September 10, 2011**, and hereby recommended that it be accepted in fulfilment of the design requirements for the degree in **Bachelor of Science in Computer Engineering.**

Engr. Dionis A. Padilla
Panel Member

Engr. Joshua B. Cuesta
Panel Member

Engr. Analyn Yumang
Panel Member

This design report is hereby approved and accepted by the School of Electrical Engineering, Electronics Engineering, and Computer Engineering in partial fulfilment of the requirements for the degree in **Bachelor of Science in Computer Engineering.**

Felicito S. Caluyo

Felicito S. Caluyo
Dean, School of EECE

ACKNOWLEDGEMENT

It is with great pleasure that we acknowledge the efforts of those individuals who have taken part to the development of this study.

We would like to thank our adviser, Engr. Ayra Panganiban for guiding us and sharing her time and knowledge on the study. To the panel members who have allotted their time for our oral presentation and for checking our paper for the necessary revisions;

To our professor, Engr. Noel Linsangan, who tolerantly helped us with the necessary revisions needed for our paper, provided us handy guidelines and documents for the completion of this project and inspired us to strive for the betterment of our research;

To our friends and colleagues who helped and supported us with this design;

To our parents, for their unending support and encouragement; and

Above all, we humbly give our sincerest gratitude to the Almighty God for giving us the strength, patience, unfading guidance and for imparting us the wisdom to accomplish this paper.

TABLE OF CONTENTS

TITLE PAGE	i
APPROVAL SHEET	ii
ACKNOWLEDGEMENT	iii
TABLE OF CONTENTS	iv
LIST OF TABLES	vi
LIST OF FIGURES	vii
ABSTRACT	viii
Chapter 1: DESIGN BACKGROUND AND INTRODUCTION	1
BACKGROUND	1
STATEMENT OF THE PROBLEM	2
OBJECTIVES OF THE DESIGN	3
IMPACT OF THE DESIGN	3
DESIGN CONSTRAINTS	4
DEFINITION OF TERMS	5
Chapter 2: REVIEW OF RELATED DESIGN LITERATURES AND STUDIES	10
IRIS RECOGNITION TECHNOLOGY	10
IMAGE QUALITY	11
IMAGE QUALITY METRICS	11
PROXIMITY SENSOR	14
IRIS IMAGE ACQUISITION	14

IRIS RECOGNITION SYSTEM AND PRINCIPLES 15

BIOMETRIC TEST METRICS 16

Chapter 3: DESIGN PROCEDURES 19

HARDWARE DEVELOPMENT 20

SOFTWARE DEVELOPMENT 26

PROTOTYPE DEVELOPMENT 28

Chapter 4: TESTING, PRESENTATION AND INTERPRETATION OF DATA 34

SENSOR OUTPUT TEST 34

IMAGE QUALITY TEST 36

DATASETS 40

IMPACT ANALYSIS 42

Chapter 5: CONCLUSION AND RECOMMENDATION 44

BIBLIOGRAPHY 47

APPENDIX 49

APPENDIX A - Operation's Manual 49

APPENDIX B - Pictures of Prototype 57

APPENDIX C - Program Listing 58

APPENDIX D - Data Sheets 108

APPENDIX E - IEEE Article Format 124

LIST OF TABLES

Table 4.1: Proximity Sensor Settings 34

Table 4.2: Sensor Output Testing 35

Table 4.3: Camera Specifications 36

Table 4.4: Iris Image Quality Assessment 38

Table 4.5: Enrolled Captured Iris Images 40

Table 4.6:

Inter-class comparisons of Haar wavelet at Level 4 vertical coefficient 41

Table 4.7:

Intra-class comparisons of Haar wavelet at Level 4 vertical coefficient 42

LIST OF FIGURES

Figure 2.1: Iris Diagram 10

Figure 3.1: Conceptual Framework 20

Figure 3.2: Block Diagram 21

Figure 3.3: Schematic Diagram 24

Figure 3.4: System Flowchart 26

Figure 3.5: Relational Model 28

Figure 3.6: 5-V Power Supply 29

Figure 3.7: NIR LED 30

Figure 3.8: Proximity Sensor 31

Figure 3.9: Gizduino 32

Figure 3.10: Webcam 32

Figure 4.1: Selected Iris images from Engr. Panganiban's system 37

Figure 4.2: Selected Iris images from the current system 38

ABSTRACT

Biometrics is becoming popular nowadays due to its very useful security application. These technologies use the unique characteristics of an individual in an electronic system for authentication. There are numbers of biometrics technology and among those; the iris recognition technology is considered the most reliable since human iris is unique and cannot be stolen. The purpose of this design is to improve an existing iris recognition system developed by Engr. Panganiban which is entitled "CCD Camera with Near-Infrared Illumination for Iris Recognition System". The proposed design aims to automate the existing iris recognition system through the use of the following materials: webcam, Gizduino microcontroller, NIR LEDs, power supply, and a proximity sensor. The NIR LEDs, which illuminates the iris, were placed in a circular case attached in the webcam. The iris image that would be captured in this design would only produce little noise since the light produced by the NIR LEDs would be pointing to the pupil of the eye and thus, the iris image template would not be affected. The automation block as its name implies, automates the capturing of the webcam through the use of the sensor, that is connected to the microcontroller in which is handled by the image acquisition software. An additional feature of this design is the real-time processing of image. Once the iris was captured, the software would automatically perform iris segmentation, normalization, template encoding and template matching. It would then display if your iris is authenticated (enrolled) or not. In matching the templates, when the Hamming distance value is greater than or equal to 0.1060, the iris templates do not match but when the HD value is less than 0.1060, the iris template are from the same individual. In comparing the accuracy of the iris templates in our design, the Degrees-of-Freedom (DoF) was computed. The computed DoF of our design is 80, which is higher than that of Engr. Panganiban's work.

Keywords: biometrics, iris recognition, hamming distance, wavelet, real-time-image processing.

Chapter 1

DESIGN BACKGROUND AND INTRODUCTION

BACKGROUND

Biometrics is becoming popular nowadays due to its very useful security applications. The technology uses the unique characteristics of an individual in an electronic system for authentication. Biometric technologies, used as a form of identity access management and access control, are becoming the foundation of an extensive array of highly secure identification and personal verification solutions. There are several of applications for biometrics which include civil identity, infrastructure protection, government/public safety and the like. As for the main intention of this design is to implement it for security function since it is very useful to this field having a fact that an iris of a human is the most unique, even for a person, the left iris has different pattern of wavelets compared to that of the right iris of the same person. This design includes an automated CMOS camera and proximity sensor for iris recognition system. A CMOS camera, or complementary metal oxide semiconductor camera, has a CMOS image sensor in which has an ability to integrate a number of processing and control functions. These features include timing logic, exposure control, white balance and the likes. The proximity sensor automates the camera. The sensor decides on whether the target is positioned for capture. The required input information is the iris image of a person for the iris recognition system database. The image

will be processed and analyzed by the built-in algorithm in MATLAB. The iris image will be stored in the database as stream of bits. These bits will serve as the identification of the person who enrolled it and will also be used for template matching, a process of finding the owner of the iris template by comparing every iris template in the database.

STATEMENT OF THE PROBLEM

The existing Image Acquisition of the Iris Recognition System developed by Panganiban (2009), entitled "CCD Camera with Near-Infrared Illumination for Iris Recognition System" recommends the enhancement of the device to improve the performance of the system. The purpose of this innovation is to answer the following questions:

1. Since quality image affects the critical success of iris image enrolment. What camera should be used to get a better quality image to get a clear detail of the captured iris image?
2. What are the additional components and changes needed, and how can an installation of proximity sensor automate and enhance the precision of the camera and improve the matching rate of accuracy?

OBJECTIVES OF THE DESIGN

The primary objective of this design is to automate and improve the existing Image Acquisition of the Iris Recognition System by Engr. Panganiban. Specifically, for the success of this design, the following objectives must be met;

1. The Camera to be used, with the help of the NIR LEDs, must be able to produce an image of the subject's iris.
2. NIR LEDs must be located where it would give enough IR light to the subject's iris. This would help make the iris more visible to the camera and to the image for capture.
3. The Proximity sensor should be installed to the system which would detect whether the person is at the correct distance and position before capturing the subject's iris.
4. The system must be able to recognize the difference between the irises to be processed through Hamming distance values and show the separation of classes through degree-of-freedom (DoF).
5. The system must have a DoF improvement on Engr. Panganiban's design.

IMPACT OF THE DESIGN

The design is an Automated Iris Recognition System; it is generally made for improving its image acquisition. This would capture an image of the iris. Nowadays, this biometric technology shows an increasing promise on the

security system for it studies the unchanging measurable biological characteristics that are unique to each individual. Among the existing biometric devices and scanners available today, it is generally conceded that iris recognition is the most accurate. The design can be used as a prototype which can be implemented by companies, governments, military, banks, airports, research laboratories, border control for security purposes for allowing and limiting access to a particular information or area. The government officials could also use this design for identifying and recording information of individuals and criminals.

Iris recognition technology can be used in places demanding high security. Physical access-based identification, which includes anything requiring a password, personal identification number or key for building access or the like, could be replaced by this technology. Unlike those physical methods of identification, human iris cannot be stolen. This technology addresses the problems of both password management and fraud.

DESIGN CONSTRAINTS

Good quality iris image can only be produced if the eye is approximately 3 to 4 cm away from the camera. A solid red light from the proximity sensor would indicate that the human eye is within the range of 4 to 5 cm. Every time an object is sensed, the red LED generates a solid light and the camera captures an

image of the object. The system does not involve iris image processing and matching of individuals with eye disorders or contact lenses. Since with these situations, the iris image will be affected. Also, the system will only work properly when the captured image is an iris otherwise it will result to an error. The speed of the system is limited by the computer specifications where the software is deployed. The recommended system requirements for the software application is a multi-core 2.20 GHz or higher for the CPU, a 4.00 GB or higher for the RAM and Windows 7 for the operating system.

DEFINITION OF TERMS

Authentication – the process of determining whether someone or something is enrolled in the system, or has authorized to be.

Biometrics – the science and technology of measuring and analyzing biological data; refers to technologies that measure and analyze human characteristics, such as fingerprints, eye retinas and irises, voice patterns, facial patterns and hand measurements, for authentication purposes.

Camera – a device that converts images into electrical signals for television.

CMOS / Complementary Metal-Oxide Semiconductor - a semiconductor technology used in the transistors that are manufactured into most of microchips.

Database – the collection of data on computer; a systematically arranged collection of computer data, structured so that it can be automatically retrieved or manipulated.

De-noising – the extraction of a signal from a mixture of signal and noise.

Enrolment – the process of putting something on a database for the first time.

Focus – the point where rays of light, heat, etc. or waves of sound come together, or from which they spread or seem to spread; specifically, the point where rays of light reflected by a mirror refracted by a lens meet or the point where they would meet if prolonged backward through the lens or mirror.

Hamming distance – the difference between letter or number sequences: a measure of the difference between two words or messages, expressed by the number of characters needing to be changed in one message to obtain the other.

Hardware – physical components of a computer system.

Illumination – an act of illuminating; the provision of light to make something visible or bright, or the fact of being lit up.

Image – a picture, idea, or impression of a person, thing, or idea; or a mental picture of a person, thing, or idea.

Image acquisition – image processing, the alteration or manipulation of images that have been scanned or captured by a digital recording device.

Image capture – employing a device, such as a scanner, to create a digital representation of an image.

Image quality – used to refer to the degree of visibility of relevant information in an image.

Infrared – electromagnetic radiation having a wavelength just greater than that of red light but less than that of microwaves, emitted particularly by heated objects.

Iris – the pigmented, round, contractile membrane of the eye, suspended between the corneas and perforated by the pupil; regulates the amount of light entering the eye.

Iris Recognition – a type of pattern recognition of a person's iris recorded in a database for future attempts to determine or recognize a person's identity when the eye is viewed by a reader.

MATLAB / Matrix Laboratory – a high-level programming language for technical computing from The MathWorks, Natick, MA; used for a wide variety of scientific and engineering calculations, especially for automatic control and signal processing. It has an interactive environment that enables you to perform computationally intensive tasks faster than with traditional programming language such as C, C++, and Fortran.

Normalization – the process of efficiently organizing data in a database.

Proximity sensor – a sensor that can detect the presence of nearby objects without any physical contact.

Wavelets – a wave-like oscillation with amplitude that starts out at zero, increases, and then decreases back to zero; a waveform that is bounded in both frequency and duration.

Sensor – a device, such as a photoelectric cell, that receives and responds to a signal or stimulus.

Segment – the part into which something is divided.

Segmentation – the process of partitioning a digital image into multiple segment; in this case, the process of locating the iris region.

Software – a collection of computer programs and related data that provide the instructions telling a computer what to do and how to do it.

CHAPTER 2

REVIEW OF RELATED DESIGN LITERATURES AND STUDIES

Iris Recognition Technology

Biometrics became popular in security applications due to its personal identification and verification based on the physiological and behavioural characteristics of the subject. Among the existing biometric technologies, it is iris recognition that is considered promising which uses the apparent pattern of the human iris (Panganiban, 2010). The iris is a muscle within the eye that regulates the size of the pupil which controls the amount of light that enters the eye. It is the colored portion of the eye with coloring based on the amount of melatonin pigment within the muscle. The coloration and structure of the iris is genetically linked but the details of the patterns are not (National Science and Technology Council, 2006).

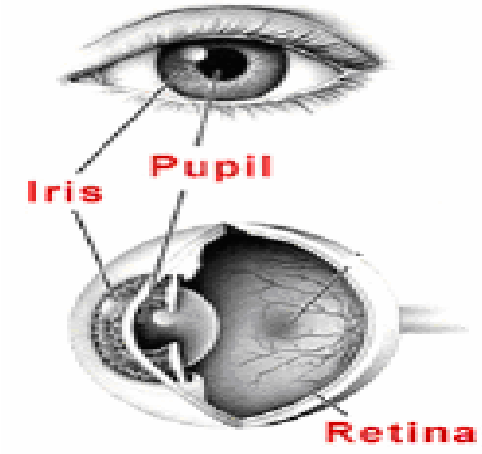


Figure 2.1 Iris Diagram

Irises contain approximately 266 distinctive characteristics, about 173 of which are used to create the iris template and serves as a basis for biometric identification of individuals. Iris patterns possess high inter-class dependency, and low intra-class dependency (Daugman, 1993).

Image Quality

According to Kalka, et al., the performance of the iris recognition system, particularly recognition and segmentation, and the interoperability are highly dependent in the quality of the iris image. There are different factors that affect the image quality namely defocus blur, motion blur, off-angle, occlusion, lighting, specular reflection, and pixel-counts.

The camera must possess excellent imaging performance in order to produce accurate results. In a CMOS (Complementary Metal Oxide Semiconductor) Image sensor, each pixel has its own charge-to-voltage conversion. CMOS image sensor often includes amplifiers, noise-correction, and digitalization circuits, so that the chip outputs digital bits. Because of these features, the design complexity increases and the area available for light capture decreases.

Iris Image Quality Metrics

Iris Image Quality Document, in Part 6 of ISO/IEC 29794, establishes terms and definitions that are useful in the specification, characterization and test of iris image quality. Some of the common quality metrics for iris images are the

following: Sharpness, Contrast, Gray scale density, Iris boundary shape, Motion blur, Noise and Usable Iris Area.

Sharpness is the factor which determines the amount of detail an image can convey. It is affected by the lens, particularly the design and manufacturing quality, focal length, aperture, and distance from the image center, as well as the sensor (pixel count and anti-aliasing filter). In the field, sharpness is affected by camera shake, focus accuracy, and atmospheric disturbances like thermal effects and aerosols. Lost sharpness can be restored by sharpening, but sharpening has limits. Over sharpening can degrade image quality by causing halos to appear near contrast boundaries.

Dynamic range (or exposure range) is the range of light levels a camera can capture, usually measured in f-stops, Exposure Value, or zones. It is closely related to noise: high noise implies low dynamic range.

Contrast, also known as gamma, is the slope of the tone reproduction curve in a log-log space. High contrast usually involves loss of dynamic range — loss of detail, or clipping, in highlights or shadows.

Motion blur is the apparent streaking of rapidly moving objects in a still image or a sequence of images. This results when the image being captured changes during the grabbing of a single frame, either due to rapid movement or long exposure.

Pixel resolution is often used for a pixel count in digital imaging. An image of N pixels high by M pixels wide can have any resolution less than N lines per

picture height, or N TV lines. But when the pixel counts are referred to as resolution, the convention is to describe the pixel resolution with the set of two positive integer numbers, where the first number is the number of pixel columns (width) and the second is the number of pixel rows (height), for example as 640 by 480. Another popular convention is to cite resolution as the total number of pixels in the image, typically given as number of megapixels, which can be calculated by multiplying pixel columns by pixel rows and dividing by one million.

According to the same standards, the number of effective pixels that an image sensor or digital camera has is the count of elementary pixel sensors that contribute to the final image, as opposed to the number of total pixels, which includes unused or light-shielded pixels around the edges.

Image noise is the random variation of brightness or color information in images produced by the sensor and circuitry of a scanner or digital camera. Image noise can also originate in film grain and in the unavoidable shot noise of an ideal photon detector. It is generally regarded as an undesirable by-product of image capture. According to Makoto Shohara, noise is dependent on the background color and luminance. They conducted subjective and quantitative experiments for three noise models, using a modified grayscale method. The subjective experiment results showed the perceived color noise depends on the background color, but the perceived luminance noise does not.

Proximity Sensor

A proximity sensor detects the presence of nearby objects without any physical contact. This type of sensor emits a beam of electromagnetic radiation, such as infrared, and looks for changes in the field or a return signal. The proximity sensor automates the camera by deciding on whether the target is positioned for capture.

Iris Image Acquisition

Image acquisition depends highly on the image quality. According to Dong, et al. (2008), the average iris diameter is averagely 10 millimeters, and the required pixel number in iris diameter is normally more than 150 pixels in iris image acquisition systems. The International standard regulates that 200 pixels is of "good quality", 150-200 is "acceptable quality" and 100-150 is "marginal quality". The iris image with a smaller pixel is considered as of a better quality image and a bigger pixel as of less quality image.

In Panganiban's study (2010), it was mentioned that Phinney and Jelinek have claimed that near-infrared illumination is safe to the human eye. Derwent Infrared Illuminators supported the safeness of near-infrared illumination to the eye. Studies showed that filtered infrared is approximately 100 times less hazardous than the visible light.

Iris Recognition System and Principles

Libor Masek's proposed algorithm showed an automatic segmentation algorithm which localise the iris region from an eye image and isolate eyelid, eyelash and reflection areas. The circular Hough transform, which localised the iris and pupil regions, was used for the automatic segmentation and the linear Hough transform was used for localising occluding eyelids. Thresholding was performed for the isolation of the eyelashes and reflections. The segmented iris region was normalised by implementing Daugman's rubber sheet model. The iris is modelled as a flexible rubber sheet, which was unwrapped into a rectangular block with constant polar dimensions to eliminate dimensional inconsistencies between iris regions. Then the features of the iris were encoded by convolving the normalised iris region with 1D Log-Gabor filters and phase quantising the output in order to produce a bit-wise biometric template. The Hamming distance was chosen as a matching metric. This gave a measure on the number of bits that disagreed between two templates. A failure of statistical independence between two templates would result in a match. This means that the two templates were considered to have been generated from the same iris if the Hamming distance produced was lower than a set Hamming distance.

In the proposed algorithm of Panganiban (2010), the feature vector was encoded using Haar and Biorthogonal wavelet families at various levels of decomposition. Vertical coefficients were used for implementation because of the dominant features of the normalized images that were oriented vertically.

Hamming distance was used to define the inter-class and intra-class relationships of the templates. The computed number of degrees of freedom which was based on the mean and the standard deviation of the binomial distribution demonstrated the separation of iris classes. Proper choice of threshold value is needed in the success of the iris recognition. But if there were instances where a clear decision cannot be made based on a preset threshold value, the comparison between the relative values of Hamming distances can lead to correct recognition. The determination of identity in her study was based on both the threshold value and on a comparison of HD values. The test metrics proved that her proposed algorithm has a high recognition rate.

Biometric Test Metrics

Ives, et al. (2005) determined the consequences of compression through the analysing the compression rate. Also, each pair of curves (False Rejection Rate (FRR) and False Accept Rate (FAR)) represents the comparison of each compressed database against the original database. An original versus original comparison is included as a baseline. The compression ratio increases, the FAR curve remains virtually unchanged, while the FRR curves move further to the right which causes an increased Equal Error Rate (EER, where FAR = FRR), and an increased number of errors (False Accepts + False Rejects) which reduces overall system accuracy.

Sarhan (2009) compares the iris images by using the Hamming distance which provides a measure as to how many bits are the same between two patterns. The number of degrees of freedom represented by the templates measures the complexity of iris patterns. This was measured by approximating the collection of inter-class Hamming distance values as binomial distribution. FAR (False Accept Rate) is the probability that the system incorrectly matches the input pattern to the non-matching template in the database. The FRR (False Reject Rate) is the probability that the system fails to detect a match between the input pattern and a matching template in the database. The ROC (Relative Operating Characteristic) plot is the visual characterization of the trade-off between the FAR and FRR. The EER (Equal Error Rate) is the rate at which both accept and reject errors are equal.

Panganiban (2010) determined the performance of each feature of the vector in terms of the accuracy over vector length. The threshold values were identified through the range of the Hamming distance. Poor Quality means that the Hamming distance value is 10 % lower than the threshold value. Moderate Quality means that the user has to decide whether the Hamming distance value agrees with the desired result. This occurs when the value is ± 10 % of the threshold values. Good Quality means that the Hamming 40 distance value is 10% higher than the threshold value. False Accept Rate (FAR) is the probability that the system accepts an unauthorized user or a false template which is computed using the formula $FAR = P_{inter}/n$, where P_{inter} is the number of HD

values that fall under Poor Quality of the inter-class distribution and n is the total number of samples. False Reject Rate (FRR) is the probability that the system rejects an authorized user or a correct template which is computed using the formula $FRR = P_{intra}/n$, where P_{intra} is the number of HD values that fall under Poor Quality of the intra-class distribution and n is the total number of samples. The Equal Error Rate (EER) compares the accuracy of devices. The lower the EER, the more accurate the system is considered to be. The characteristic of the wavelet transform are the concept used in encoding iris bit patterns. These metrics are useful in achieving the accuracy and efficiency of wavelet coefficients.

Chapter 3

DESIGN PROCEDURES

The design is an automated iris recognition system with a hardware that consists of a webcam, Gizduino microcontroller, NIR LEDs, power supply, and a proximity sensor. Figure 3.1 illustrates the conceptual framework of the design. The proximity sensor sense objects that are in front of its transceiver, in the design, the face of the person is the target of the proximity sensor. When the target is within the detecting range of the sensor, the sensor will output a signal that is treated as an input to the microcontroller, and this will command the webcam to capture an image. Through proper alignment, this captured image would be the eye of the subject. The NIR light serves as the illuminations to acquire iris of the eye visible to the webcam. After the webcam captures the eye, the image acquisition software produces the iris image that will be sent to the iris recognition algorithm for analysis.

Hardware Development

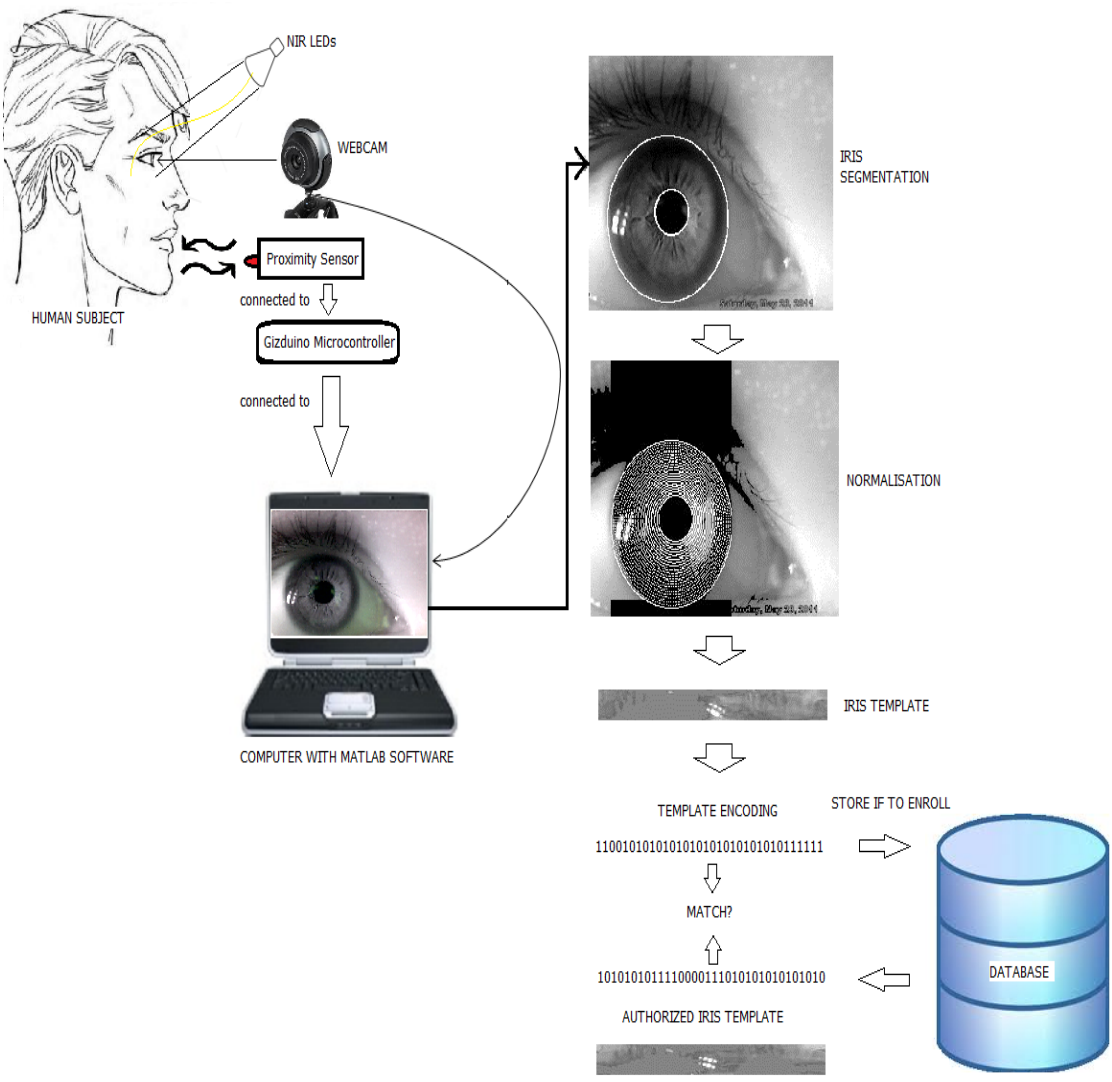


Figure 3.1 Conceptual Framework

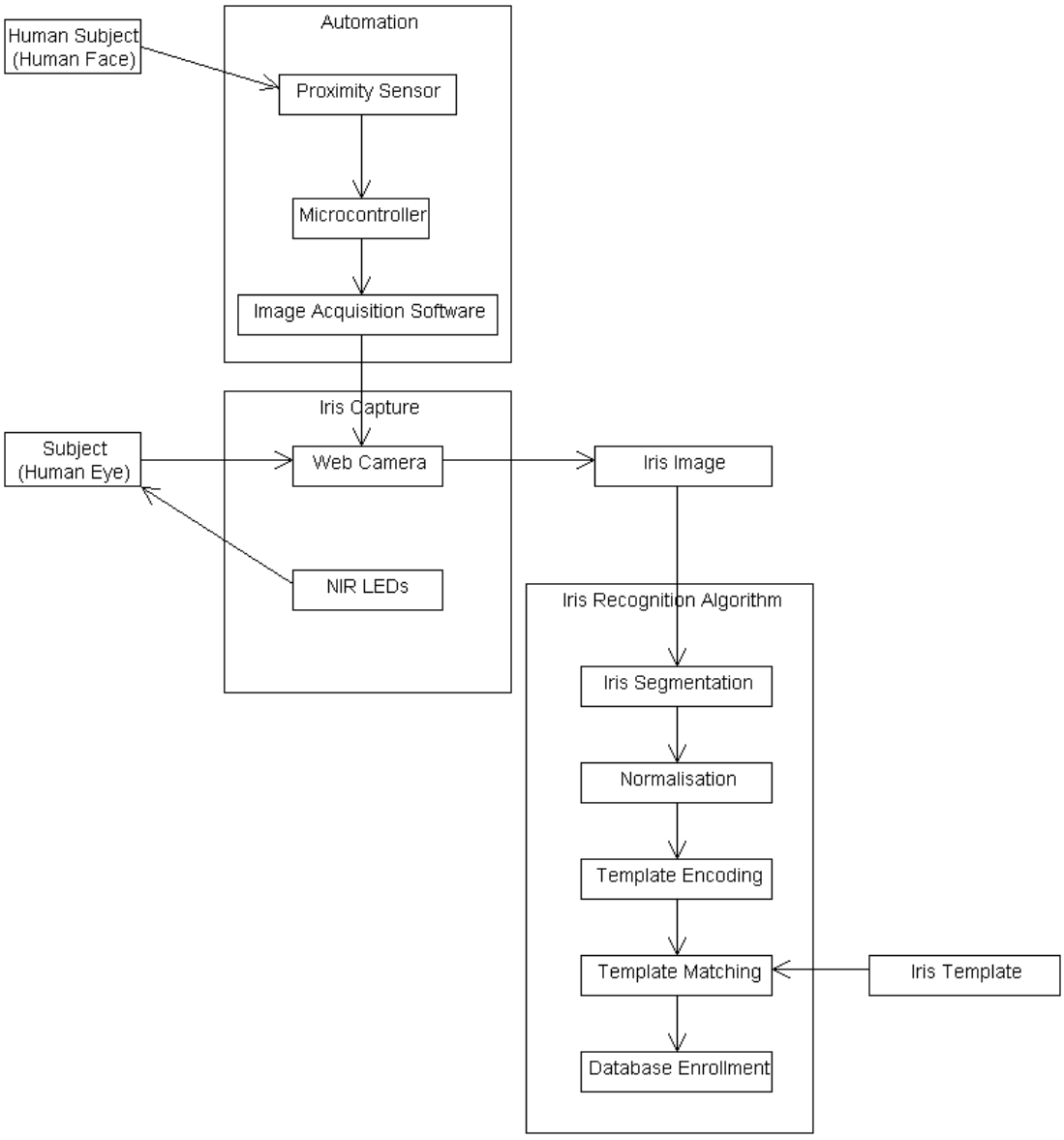


Figure 3.2: Block Diagram

Figure 3.2 represents the block diagram that was implemented to attain the goals of the design. The automation part is composed of the proximity sensor, the microcontroller and the image acquisition software. This automation block as its name implies, automates the capturing of the webcam through the use of the sensor, that is connected to the microcontroller in which is handled by the image

acquisition software. The proximity sensor senses objects within 10cmrange from its transceiver. The microcontroller used is the Gizduino microcontroller manufactured and produced by E-Gizmo. The image acquisition software is developed using MATLAB R2009a. The next part is the Iris Capture block. It consists of the webcam and the NIR LEDs. The webcam is connected to the computer through its USB cord. The NIR LEDs are the one responsible for the visibility of the iris to the webcam. If the image acquisition software tells the webcam to capture, the webcam will do so and an iris image will be produced. The final part is the iris recognition algorithm. The iris recognition algorithm starts with the iris segmentation process. It is based on the circular Hough transform which is similar to the equation of a circle ($X_C^2 + Y_C^2 = r^2$). Since the iris of the eye is ideally shaped like a circle, the Hough transform is used to determine the properties of geometric objects found in an image like circles, and lines. Canny edge detection is used to detect edges of shapes. It is developed by John F. Canny in 1986. Horizontal lines are drawn on the top and bottom eyelid to separate the iris and two circles are drawn, one for the pupil and the other one for the iris. The value of the iris radius to be used ranges from 75 to 85 pixels and for the pupil radius ranges from 20 to 60 pixels. After the iris is segmented, it is normalized. In normalization, the segmented iris is converted to a rectangular shaped-strip with fixed dimensions. This process uses Daugman's rubber sheet model. The image will then be analyzed using 2D wavelets at maximum level of 5. After that, a biometric template is produced. Similar to

Engr. Panganiban's work, the wavelet transform is used to extract the discriminating information in an iris pattern. Only one mother wavelet is used which is the Haar because it produced the highest CRR according to Engr. Panganiban's thesis. The template is encoded using the patterns that yielded during the wavelet decomposition. Then, the algorithm will check if the template matches another template stored in the database by using its binary form to compute for the hamming distance of the two templates. This is done by using the XOR operation. A template can also be added to the database by using MS SQL queries.

Figure 3.3 describes the schematic diagram of the hardware components used in the design project. The Near Infrared LEDs are powered by the 5V power supply. The power supply is composed of a transformer, rectifier, capacitor and a regulator. The transformer converts electricity from one voltage to another with minimal loss of power. It only works with an alternating current because it requires a changing magnetic field to be created in its core. Since 5-V supply is only needed, step-down transformer was used. The voltage source was reduced to 12-V AC. The rectifier converts an AC waveform into a DC waveform. It uses diodes which allows current to flow through it in one direction. The Full-Wave Rectifier converted 12-V AC to 12-V DC. The electrolytic capacitor smoothen the ripple voltage formed in the rectification process. The regulator makes the voltage stable and accurate. A heat sink was attached to dissipate the heat produced by the circuit.

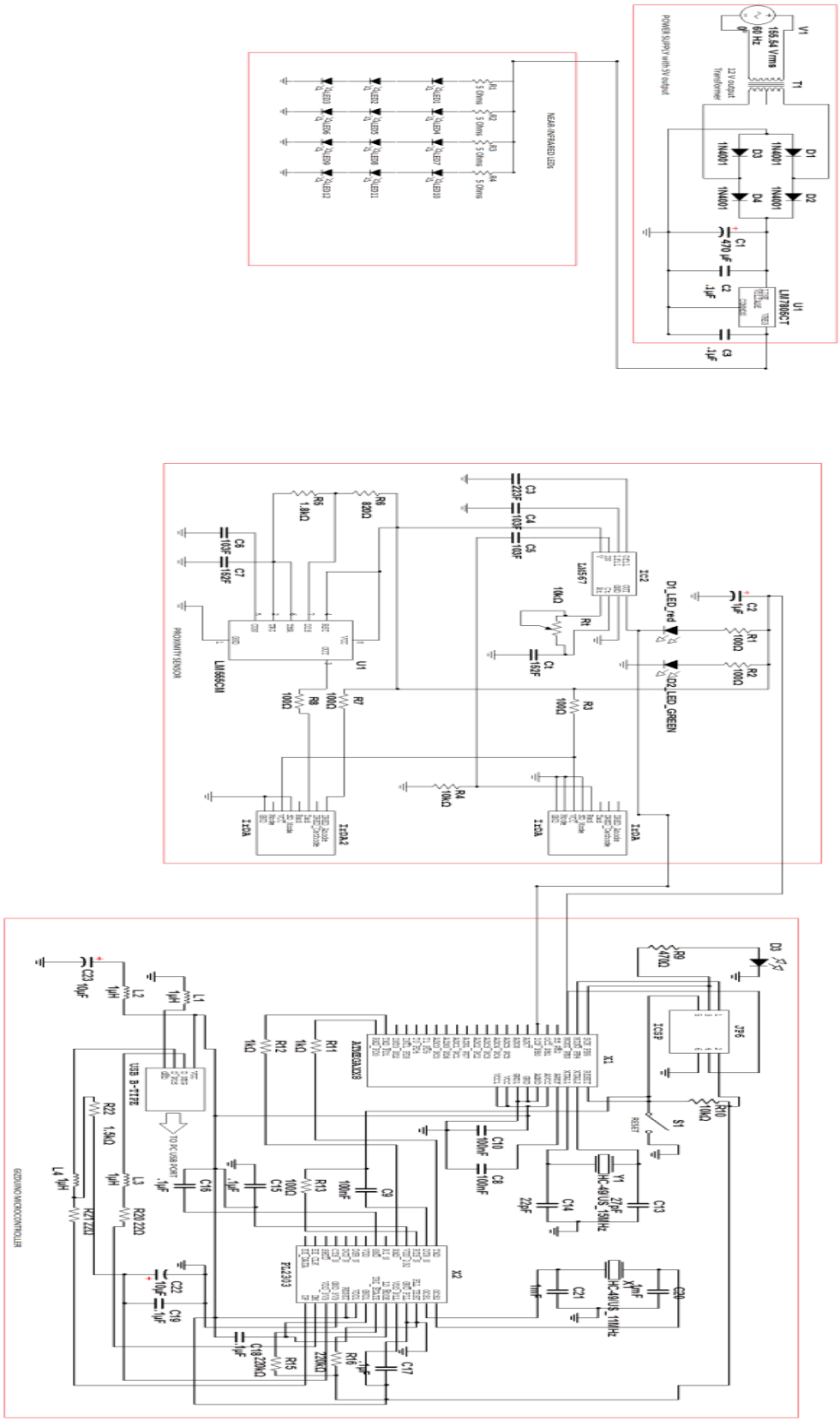


Figure 3.3: Schematic Diagram

The Near Infrared LEDs serves as the lighting source. The light produced by the near-infrared diodes is only visible in the camera and not with the human eye. It produces less noise in the image when captured than visible light. The resistors used each have 5-ohms resistance. This was computed using the formula:

$$R = (V_S - V_F) / I_F$$

where V_S is the voltage source of 5-V, V_F is the voltage drop of 1.5-V and an I_F is a current of 100-mA. The formula would produce a resistance of 35-ohms. But considering that we are to connect in parallel four rows of 3 NIR LEDs in series, the resulting resistance value 'R' connected in series with the 3 NIR LEDs on each row would be 5-ohms.

The proximity sensor detects the presence of nearby objects without any physical contact. This type of sensor emits a beam of electromagnetic radiation, such as infrared, and looks for changes in the field or a return signal. This gives the appropriate signal to the image-capturing software when the subject is in the right position for iris image acquisition.

The Gizduino microcontroller is a clone of Arduino microcontroller made by the company E-Gizmo. It has a built-in ATMEGA microcontroller and PL2303 USB to RS-232 Bridge Controller.

Software Development

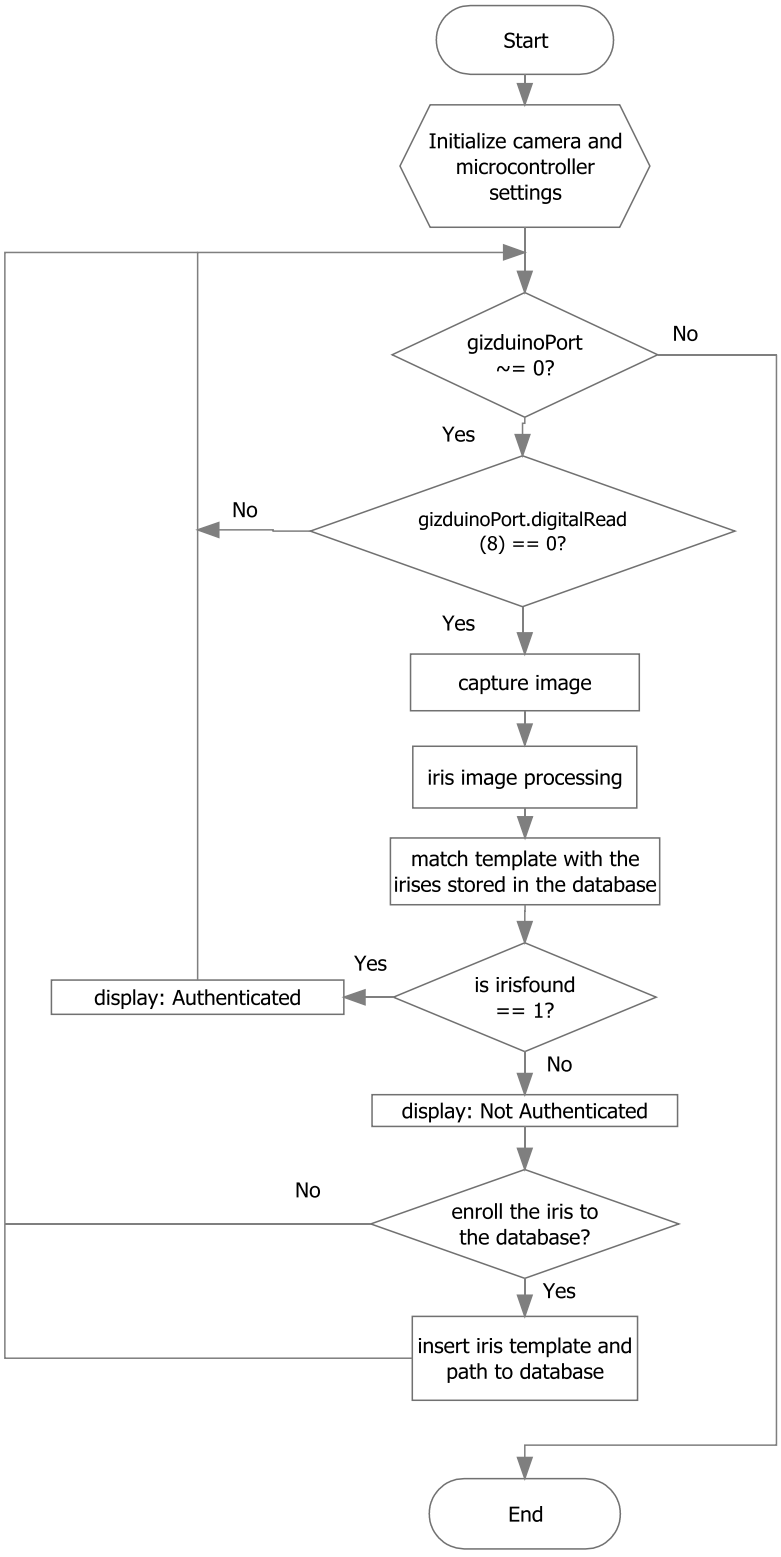


Figure 3.4: System Flowchart

Figure 3.4 illustrates the flowchart of the system. First, the system initializes the camera and the microcontroller settings. Then, it checks whether the Gizduino microcontroller is connected or not by checking the value of gizduinoPort. While it is equal to zero, the system will end its process. But while its value is not equal to zero, meaning the MCU is still connected, it inspects if the person's face is within the correct distance by checking the value of gizduinoPort.digitalRead(8). If the value is zero, it means that the distance is correct according to the proximity sensor and the program triggers the camera to capture the iris image. After capturing the image, the system processes it, extracts the iris feature and encodes the template into bits. After that, the system compares the encoded template with all the templates stored in the database. When a match is found, the program displays a message box telling that the person's iris is authenticated and is registered on the database and then the system prepares for the next capture by going back to the distance inspection. But when it's not found, the program displays a message box again however telling that it is not found and it's not authenticated. Also, the system asks if the unauthenticated iris template is to be enrolled in the database or not. If it would be enrolled, then the iris template and its path are inserted into the database and then the system goes back to the distance inspection. Else if it's not to be enrolled, then the system just goes back to the distance inspection.

IrisDataBankDesign			
Column Name	**Data Type**	**Key Type**	**Allow Null**
IrisId	Int	PK	No
IrisPath	varchar(50)	NONE	Yes
IrisTemplate	varchar(MAX)	NONE	Yes

Figure 3.5 Relational Model

The template bits are stored in a database using Microsoft SQL 2005 Express edition. In Fig. 3.5, the IrisId field is set to auto-increment by 1 and the primary key. While the IrisPath and IrisTemplate depends on the output of the system which is inserted to the database.

Prototype Development

The design prototype has both hardware and software components. The hardware components are comprised of a 5-V power supply, Near Infrared LEDs, CMOS webcam, proximity sensor, Gizduino microcontroller and a personal computer. And for the software, the MS SQL 2005 Express Edition, MATLAB 7.8, and Arduino compiler are used. The design is assembled in a way that the subject's eye would be captured is aligned with the camera lens with respect to the time the sensor detects that the subject's face is on the specific range, and sends signal to the microcontroller to automate the camera for capturing the iris image.

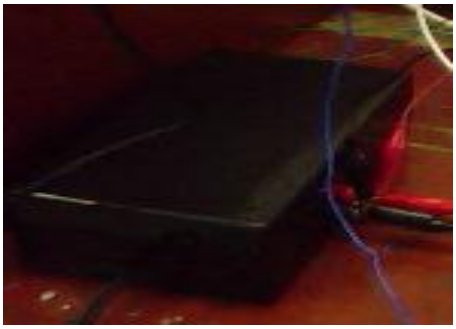


Figure 3.6: 5V Power Supply

5V Power Supply

In the hardware part, a 5-V 750-mA power supply is used to power up the NIR LEDs. It is composed of a transformer, rectifier, capacitor and a regulator. The transformer used is a step down transformer with a turn's ratio of approximately 18.33 in which it is able to produce a secondary AC voltage of 12-V from a primary AC voltage of 220-V. The type of rectifier used is a bridge rectifier.Four 4N001 diodes are used to build it so that it produces a full-wave rectification in which the 12-V_{AC} is converted to a 12-V_{DC}. However, this produces a varying DC output. A 470-uF electrolytic capacitor is used to eliminate this and produce a small ripple voltage. To produce a 5-V DC output, a 5-V voltage regulator is used; in this case, LM7805 IC is used. This also makes the voltage stable and accurate and a heat sink was attached to it in order to dissipate the heat produced by the circuit.

Figure 3.7: NIR LED

NIR LEDs

For the NIR LEDs, a series-parallel circuit connection is used. Considering the current, each LED has a forward current of 100-mA and the power supply could only produce 750-mA output. Also taking in to account for the voltage, the typical forward voltage of each LED which are used is 1.5-V and the power supply can only produce a 5-V output. Because of these current and voltage settings, only up to 7 parallel set of LEDs and each set contains 3 IR LEDs respectively. A resistor should be used in order to protect the LEDs from burning. In the computation, 5-Ohms resistor is calculated. This value can protect the LEDs from burning because it can control the current below 100mA. Using the next lower resistor value would destroy the LEDs.

Figure 3.8: Proximity Sensor

Proximity Sensor

The proximity sensor used is an infrared proximity-collision sensor. It is produced and manufactured by E-Gizmo Electronics and Robotics Shop. It has two wires for input and one for output. The input wires which are colored red and green are for the 5V supply and the ground connection, respectively. For its power, it uses the Gizduino microcontroller board for its 5V source since this microcontroller could deliver such a voltage output. It uses a TFDU6103 IrDa transceiver (see datasheets for info). There are two of it in the sensor; one serves as the receiver and the other as the transmitter. A blockade within 10cm range will have it output a low signal. This sensor is used to send signal to the microcontroller for the program to allow the camera to take a picture whenever the sensor would detect that the person's iris to be captured is within the correct distance. Furthermore, the sensor is composed of capacitors, resistors, LM555 IC, and LM567 IC.

Figure 3.9: Gizduino

Gizduino Microcontroller

The Gizduino Microcontroller has 14 digital input/output ports and 8 analog input/output ports. The output port of the proximity sensor is connected to one of its digital input/output ports. It also has 3 ground pins, 5-V pin, and 3.3-V pin.

Figure 3.10: Webcam

Webcam

The camera used is the a4tech PK 710mj live messenger 5M Webcam. It is connected to the USB port of the computer. A manual focused camera was used so that the correct distance of the person's iris to the lens of the camera may be set specifically in a way the eye of the person would only be captured by the

camera, the focus range is set about 4cm. In this case, the image being captured by the camera is stable in terms of how far the eye is from the camera to distinguish accurately the iris to be segmented and recognized.

Camera Specification:

- Image Sensor: ¼ CMOS, 640x480pixels
- Frame Rate: 15fps: @640x480, @600x800, 30fps: @320x240,@160x120
- Lens F=2.2, f=4.6mm
- View Angle: 65 degree
- Focus Range: Manual Focus, 2cm to infinity
- Exposure Control: Automatic
- Still Image Capture Res.: 2560x2048, 1600x1280, 2000x1600, 1280x960, 600x800, 640x480, 352x288, 320x240, 160x120
- Flicker Control: 50 Hz, 60Hz and None
- Computer Port USB 2.0 port

CHAPTER 4

TESTING, PRESENTATION, AND INTERPRETATION OF DATA

Automated CMOS Camera for iris recognition through proximity sensor focus on its objective of improving an existing image acquisition of the iris recognition system developed by Engr. Panganiban and the design's automation. In this chapter, the researchers conduct experiments to identify whether the hardware and software design meet the criteria for an effective iris recognition system. Several observations and assessments are provided, together with reliable measurements or data that will support the researcher's remarks.

SENSOR OUTPUT TEST

The proximity sensor automates the system by detecting whether the person is at the correct distance and position before capturing the subject's iris. Further testing on the proximity sensor was done because there has been a suspected glitch found on the proximity sensor.

Table 4.1: Proximity Sensor Settings

SETTINGS	
Position:	Placed on top of the camera
Input:	Person's forehead

Table 4.2: Sensor Output Testing

DISTANCE(cm)	Red LED Status (Output)
1	Solid Red Light
2	Solid Red Light
3	Solid Red Light
4	Solid Red Light
5	Flickering Red Light
6	No light
7	No light
8	No light
9	No light
10	No light

As seen in Table 4.2, the correctness of the distance and position was seen on the red LED's intensity with respect to the settings indicated in table 4.1. A solid red light was seen when an object is 0cm to 4m away from the IrDA. But a flickering red light was seen when the range is within the range of 4cm to 5cm. The LED does not produce light when the object is greater than 5cm. Also, these findings were relevant to the behaviour of the camera. When the red LED has a solid light, the camera captures every time an object is sensed.

IMAGE QUALITY TEST

The performance of the iris recognition system, particularly recognition and segmentation, and the interoperability are highly dependent in the quality of the iris image.

Table 4.3: Camera Specifications

Specifications	CCD Camera	A4tech PK 710mj live messenger 5M Webcam
Image Sensor	CCD image sensor with validity pixel of PAL: 512x528/- 512x492	CMOS image sensor, 640x480pixels
Focus Range	Manual focus according to user requirement	Manual Focus, 2cm to infinity (according to user requirement)

Our group replaced Eng'r. Panganiban's CCD Camera with a CMOS Camera. The camera must possess excellent imaging performance in order to produce accurate results. In a CCD (Charge Couple Device) sensor, every pixel's charge is transferred through a very limited number of output nodes to be converted to voltage, buffered, and sent off-chip as an analog signal. All of the pixel can be devoted to light capture, and the uniformity of the output is high. In a CMOS (Complementary Metal Oxide Semiconductor) sensor, each pixel has its own charge-to-voltage conversion, and the sensor often includes amplifiers, noise-correction, and digitalization circuits, so that the chip outputs digital bits. With these, the design complexity increases and the area available for light capture decreases. The uniformity is lower because each pixel is doing its own

conversion. Also, both cameras that were used were manual focus, for the user to adjust it to their system's requirements.

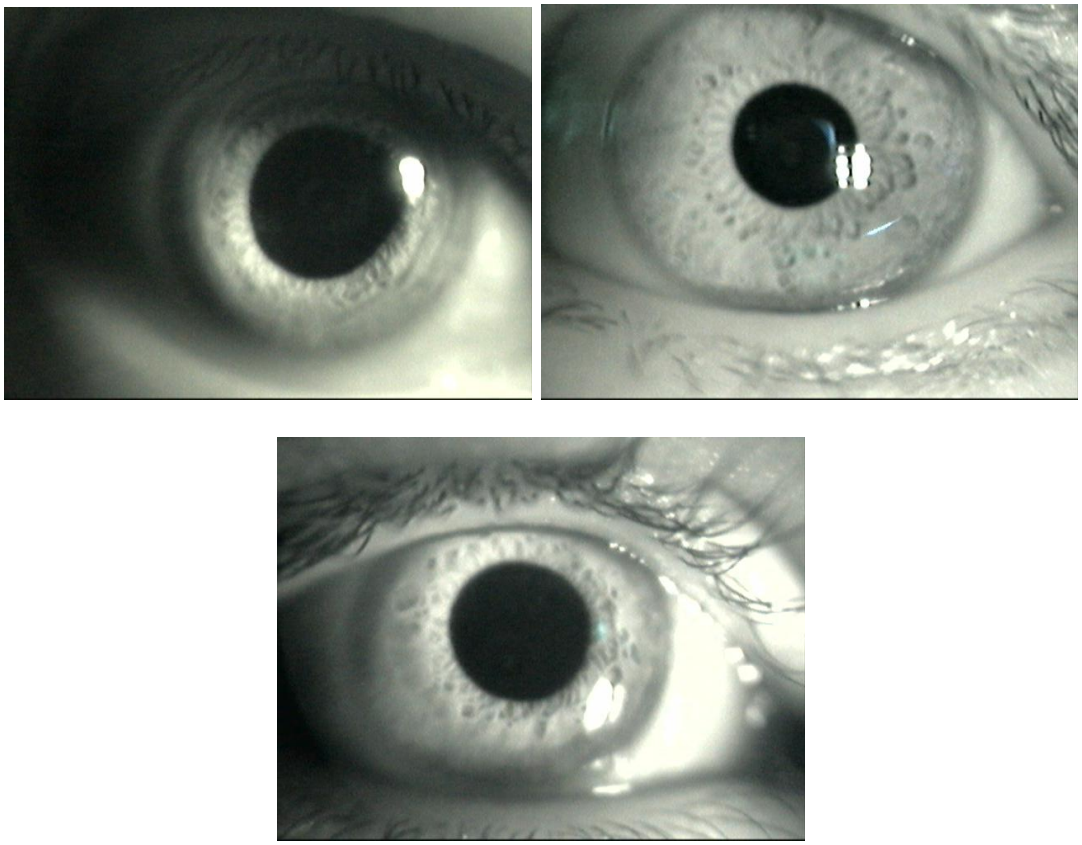


Figure 4.1: Selected iris images from Engr. Panganiban's system

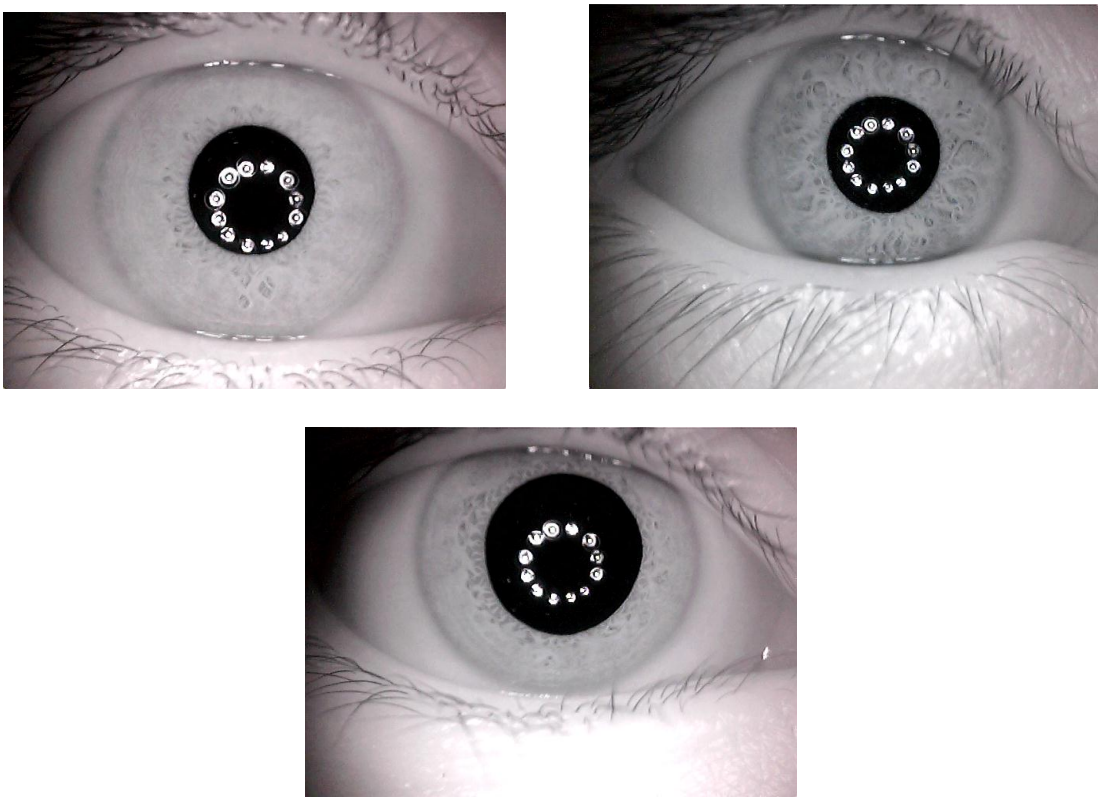


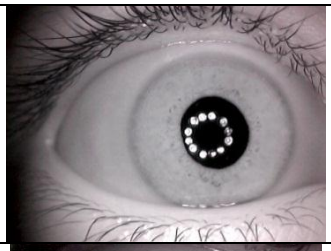
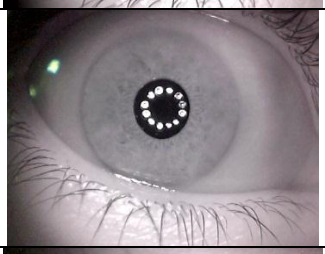
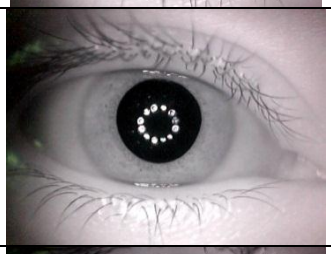
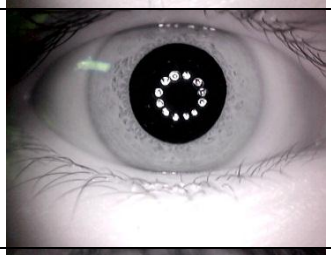
Figure 4.2: Selected iris images from the current system

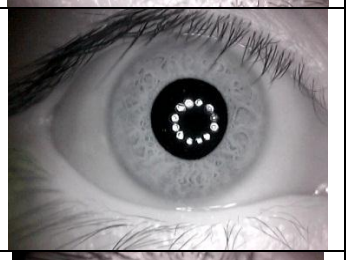
Table 4.4: Iris Image Quality Assessment

Common Quality Metrics	Figure 4.1	Figure 4.2
Blur Motion	Blurred Image	Clear Image
Noise in the Iris Image	With Noise	Without Noise
Brightness	Dark	Bright
Magnification	Blurred Image	Clear Image

In Table 4.4, it can be observed that the improved design really showed promising results. The design produced a clear and bright image even though the image was magnified in the test. The magnification testing was made by zooming in the images. Also, there was no noise in the iris image.

Table 4.5: Enrolled Captured Iris Images

ID Number	Iris Image
1	
2	
3	
4	
5	
6	

7	
8	
9	
10	

DATASETS

In Table 4.5, the iris images that were captured and enrolled into the Iris Recognition System are displayed. These images undergone image processing as discussed in the previous chapter to have its iris template be produced. The iris templates were encoded using the Haar mother wavelet because according to Engr. Panganiban's work, it resulted with the best values of Hamming distance after every iris template were compared. The Inter-class comparisons of Haar wavelet at Level 4 vertical coefficient is shown on Table 4.6. As seen on the

table, the maximum HD value is 0.1538 and the minimum is 0.1060. A zero value indicates that the iris templates are perfectly matching each other.

Table 4.6: Inter-class comparisons of Haar wavelet at Level 4 vertical coefficient

Iris Id	1	2	3	4	5	6	7	8	9	10
1	0.0000	0.1331	0.1331	0.1268	0.1060	0.1268	0.1227	0.1331	0.1081	0.1206
2	0.1331	0.0000	0.1331	0.1518	0.1351	0.1268	0.1227	0.1372	0.1372	0.1414
3	0.1331	0.1331	0.0000	0.1268	0.1351	0.1227	0.1268	0.1081	0.1247	0.1372
4	0.1268	0.1518	0.1268	0.0000	0.1247	0.1372	0.1206	0.1143	0.1227	0.1060
5	0.1060	0.1351	0.1351	0.1247	0.0000	0.1123	0.1538	0.1435	0.1351	0.1310
6	0.1268	0.1268	0.1227	0.1372	0.1123	0.0000	0.1289	0.1393	0.0977	0.1268
7	0.1227	0.1227	0.1268	0.1206	0.1538	0.1289	0.0000	0.1227	0.1102	0.1185
8	0.1331	0.1372	0.1081	0.1143	0.1435	0.1393	0.1227	0.0000	0.1123	0.1206
9	0.1081	0.1372	0.1247	0.1227	0.1351	0.0977	0.1102	0.1123	0.0000	0.1206
10	0.1206	0.1414	0.1372	0.1060	0.1310	0.1268	0.1185	0.1206	0.1206	0.0000

It is observable that when the Hamming distance value is greater than or equal to 0.1060, the iris templates do not match. In table 4.6, the Intra-class comparisons of Haar Wavelet at level 4 vertical coefficient shows that when the HD value is less than 0.1060, the iris template are from the same individual.

Using the formula for the degrees of freedom:

$$DOF = \frac{p(1-p)}{\sigma^2}$$

Where p is the mean which is equal to 0.1261 and the σ is the standard deviation which is equal to 0.03954, the number of degrees of freedom is 80. According to statistics, this is the number of degrees of freedom that the values in this case, the HD values are free to vary.

Table 4.7: Intra-class comparisons of Haar wavelet at Level 4 vertical coefficient

Iris Id	1	2	3	4	5	6	7	8	9	10
1	0.0811	0.1351	0.1310	0.1060	0.1123	0.1247	0.1227	0.1289	0.1185	0.1227
2	0.1310	0.0603	0.1019	0.1185	0.1331	0.1123	0.1268	0.1289	0.1435	0.1227
3	0.1310	0.1310	0.0686	0.1351	0.1372	0.1247	0.1476	0.1081	0.1227	0.1019
4	0.1164	0.1331	0.1497	0.0956	0.1268	0.1393	0.1247	0.1310	0.1123	0.104
5	0.1247	0.1206	0.1455	0.1247	0.0852	0.1060	0.1372	0.1351	0.1206	0.1518
6	0.1372	0.1206	0.1081	0.1289	0.1185	0.0520	0.1372	0.1227	0.0915	0.1372
7	0.1164	0.1164	0.1164	0.1080	0.1310	0.1393	0.0873	0.1185	0.1164	0.1206
8	0.1310	0.1518	0.1268	0.1310	0.1580	0.1060	0.1227	0.0748	0.1227	0.1227
9	0.1102	0.1435	0.1185	0.1019	0.1289	0.1123	0.1435	0.1164	0.0561	0.1143
10	0.1019	0.1227	0.1476	0.1143	0.1455	0.1247	0.0977	0.1247	0.1143	0.0977

IMPACT ANALYSIS

The iris recognition system of Engr. Panganiban was taken to the next level by adding real time image processing features to it. This would be easier to use for the user would just look into the camera and wait for just a short period of time for the system to capture and process his or her iris. After the image was processed, it would immediately display if the person is authenticated or not.

The designed iris recognition showed an increasing promise on the security system for it analyses the unchanging measurable biological characteristics that are unique to each individual. The design can be used as a prototype which can be implemented by in places demanding high security such as companies, governments, military, banks, airports, research laboratories and border control area. This would allow and limit access to a particular information or area. The government officials could also use this design for identifying and recording information of individuals and criminals. Physical methods of identification, which

includes anything requiring a password, personal identification number or key for building access or the like, are easily hacked or stolen but human iris cannot be stolen. This technology addresses the problems of both password management and fraud.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

CONCLUSION

Based from the results obtained, the design was proven sufficient for iris recognition. The camera used is a manual focus- CMOS camera. In a Complementary Metal Oxide Semiconductor sensor, each pixel has its own charge-to-voltage conversion, and the sensor often includes amplifiers, noise-correction, and digitalization circuits, so that the chip outputs digital bits. With these, the design complexity increases and the area available for light capture decreases. The correct positioning of the webcam, NIR LEDs and sensor produced a clearer and brighter iris image which really improves the performance of the iris recognition system. The NIR LEDs must be attached circular to the webcam so that noise that would be produced in the iris would be lessened. The light of the NIR LEDs would be directed to the pupil. Since the light reflection will be located in the pupil, it would not affect the iris segmentation and that the iris template. The case of the camera also lessens the noise since it blocks other factors that might affect the iris image and results.

The proximity sensor has a delay of 5 seconds before it sends signal for the webcam to capture the iris image. There is a delay so that the user can position his or her eye properly to the device.

Also, the results showed that when the Hamming distance value is greater than or equal to 0.1060, the iris templates do not match. The Intra-class comparison of Haar Wavelet at level 4 vertical coefficient shows that when the HD value is less than 0.1060, the iris templates are from the same individual.

From the results of the Hamming Distance in inter-class comparison, the Degrees of Freedom (DoF) computed is 80, which is higher than of Engr. Panganiban's work which is equal to 50. This shows that the comparison of iris templates in our design is more accurate.

RECOMMENDATION

Although the obtained results proved that the design is sufficient for iris recognition, the following are still recommended for the improvement of the system's performance:

1. The proximity sensor may be replaced by an algorithm such as pattern recognition that will allow the software to capture the iris image once a circular shape is near the camera.

2. The digital camera can be converted to an Infrared Camera which would replace the webcam and NIR LEDs.

3. Artificial Intelligence, such as Fuzzy Logic, can be applied to the system to improve the performance of the Iris recognition system.

4. Embedding the Iris recognition system, its hardware and software into one device can be done to have the speed of the system independent on the speed of the computer used and could also be portable.

REFERENCES

Addison, P. (2002). The Illustrated Wavelet Transform Handbook, Institute of Physics.

Bradley J., Brislawn, C., and Hopper, T. (1993). The FBI Wavelet/Scalar Quantization Standard for Gray-scale Fingerprint Image Compression. Tech. Report LA-UR-93-1659, Los Alamos Nat'l Lab, Los Alamos, N.M.

Boles, W.W. and Boashash, B.A. (1998). A human identification technique using images of the iris and wavelet transform," IEEE trans. on signal processing, vol. 46, issue 4.

Canny, J. (1986). A Computational Approach To Edge Detection, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679–714.

Cohn, J. (2006). Keeping an Eye on School Security: The Iris Recognition Project in New Jersey Schools. NIJ Journal, no. 254.

Huifang, H. and Guangshu, H. (2005). Iris recognition based on adjustable scale wavelet transform. Proceedings of the 2005 IEEE.

Kong, W. and Zhang, D. (2001). Accurate iris segmentation based on novel reflection and eyelash detection model. Proceedings of 2001 International Symposium on Intelligent Multimedia, Video and Speech Processing, Hong Kong.

Makram Nabti and Bouridane (2007). An effective iris recognition system based on wavelet maxima and Gabor filter bank. IEEE trans. on iris recognition.

Masek, L. (2003). Recognition of Human Iris Patterns for Biometric Identification.

Narote et al. (2007). An iris recognition based on dual tree complex wavelet transform. IEEE trans. on iris recognition.

Panganiban, A. (2009). CCD Camera with Near-Infrared Illumination for Iris Recognition System.

(2010). Implementation of Wavelet Algorithm for Iris Recognition System.

APPENDIX A

Operation's Manual

I. System Requirements

CPU:	Intel® Core™ i7
Memory:	4.00 GB
Operating System:	Windows 7
Software:	MATLAB R2009a

II. Installation Procedure

1. MATLAB R2009a installation (Recommended):

1.1 Load the MATLAB R2009a installer; it should automatically start the installation program whereby the first splash screen could be seen.

1.2 Agree to the Mathworks license, and then press Next.

1.3 Choose the 'Typical' installation, and then press Next.

1.4 Choose the location of the installation, and then press Next.

1.5 If the location doesn't exist, you will be prompted to create it and MATLAB will ask you for the location on where the files will be installed.

1.6 Confirm the installation settings by pressing 'Install'

1.7 MATLAB will now install, this may take several minutes

1.8 Close to the end of the installation, you will be asked if you want to set up some file associations. Choose 'Yes to All'.

1.9 After the installation has completed, you will be asked for the serial key for the software license. Enter the serial key and press Next.

1.10 MATLAB will initially make an internet connection to Mathworks. Answer 'Yes' when asked if you are a student. Then press Next.

1.11 Enter the serial key and your email address. Then press Next.

1.12 Continue with the rest of the registration process until the installation is complete.

2. Arduino Compiler installation:

(For Windows)

2.1. Get an Arduino board, and connect it to your computer with a USB cable.

2.2. Download the Arduino environment on its official website. (http://arduino.cc/en/Main/Software)

2.3. Install the drivers

2.3.1. Wait for Windows to begin its driver installation process. After a few moments, the process will fail, despite its best efforts.

2.3.2. Click on the Start Menu, and open up the Control Panel.

2.3.3. While in the Control panel, navigate to System and Security. Next, click on System. Once the System window is up, open the Device Manager.

2.3.4. Look under Ports (COM & LPT). There should be an open port named "Arduino UNO (COMxx)".

2.3.5. Right click on the "Arduino UNO (COMxx)" port and choose the "Update Driver Software" option.

2.3.6. Next, choose the "Browse my computer for Driver software" option.

2.3.7. Finally, navigate to and select the UNO's driver file, named "ArduinoUNO.inf", located in the '"Drivers" folder of the Arduino Software download (not the "FTDI USB Drivers" sub-directory).

2.3.8. Windows will complete the driver installation from there.

III. User's Manual

How to use the Gizduino microcontroller and software:

1. Connect the Gizduino microcontroller to the USB port of the computer.
2. Open the Arduino Compiler.
3. From the Menu Bar, select Tools then choose Serial Port and select the designated port where the microcontroller is connected.

4. On the Arduino workspace, enter the arduino input/output server code that is listed on Appendix C.

5. Compile the code by pressing Verify button to check for errors before uploading it to the microcontroller.

6. To upload the code to the microcontroller, press the Upload button.

7. Wait until the uploading is finished; A message "Uploading Successful" will be displayed.

How to use the MATLAB iris recognition software:

1. Open a MATLAB workspace.

2. In the directory icon, browse the folder where the source code is located, in this case the programs are stored under a folder named "Design Project".

3. In the command directory area, make sure that all files and folders are properly referenced. *Note: highlight all folders under the "Design Project" folder then right click. Choose "add to path" > "all folders and sub folders".

4. In the current directory, right click on the 'irisrecognition.m' program and choose "Run File" to run this program.

How to setup the Iris Recognition Design:

*Note: The Iris Recognition software must be properly referenced on MATLAB and the arduino code provided must be uploaded on the arduino microcontroller.

1. Plug-in the source to 220-V supply and the USB cable to the Computer or Laptop. Be sure that the computer being used complies with the design's system requirements.

2. Make sure that the arduino input/output server code is uploaded to the microcontroller, and the Iris Recognition Software is on the current directory on MATLAB.

3. Open MATLAB R2009a, highlight all folders under "Iris Recognition System folder" (Software Design) then right click. Choose "add to path" > "all folders and sub folders". Then run the Matlab program.

4. Run the MATLAB software 'irisrecognition.m' provided.

5. Adjust the position of the Camera and IR LEDs to where the subject is comfortable with; just be sure that it would capture the subject's iris image accurately. Then compile and run the MATLAB program of iris recognition system.

6. The User must move his/her head close to the camera within the proximity range 4 to 5-cm away. From here the design must perform its auto-capture and real-time process of data.

7. If the iris image captured isn't within the authenticated list on the database, the user will be asked to whether or not enrol the iris image. Otherwise the program will simply display "unauthenticated" iris image pattern.

8. After the authentication the program will go back to its status of auto-capturing an iris image.

9. To terminate, simply exit the MATLAB program.

IV. Troubleshooting Guides and Procedures

1. If there is a problem on the Arduino Connection on MATLAB

    a) Upload the adiosrv.pde on the Gizduino

    b) Check if the COM PORT where the Gizduino is connected is the same on the SerialPort() definition on MATLAB

2. If the image is blurred, check and adjust the focus of the camera. Twist its lens to have the desired focus.

3. Uploading Errors on Gizduino

    a) Check the syntax for errors.

    b) Consult the website, www.arduino.cc, for more information.

4. Unknown MATLAB function

    a) Check if the program files are located at the current directory window of MATLAB.

b) If the files are already the current directory of MATLAB, select all files and right click then add to path all the folders and subfolders.

5. If there are many cameras connected and installed on the laptop, check the image acquisition toolbox of MATLAB and select the adaptor name and device ID of the desired camera to be used.

6. There is no light emitted by the LEDs

a) Make sure that the polarity on the source to LED connection is correct.

b) Check the proper connection of the LEDs in series and parallel.

c) Plug the power supply.

V. Error Definitions

MATLAB:

1. Error in using videoinput in MATLAB – The camera device is not detected by MATLAB or its DEVICEID or adapter name is invalid.
2. Error at segmentiris.m – There are no detectable circular patterns.
3. COM PORT unavailable- There is no devices connected on the particular Serial COM port
4. Function or CD diagnostics or directory not found- make sure that the current directory in MATLAB is the one where the .m files are placed

Arduino Compiler:

1. Error Compiling- Check the syntax for errors

2. Serial Port not found- The Gizduino microcontroller is connected to a different Serial Port or there is nothing connected.

APPENDIX B

Pictures of Prototype

APPENDIX C

Program Listing

◆◆

Arduino.m

```
classdef arduino < handle

    % This class defines an "arduino" object
    % Giampiero Campa, Aug 2010, Copyright 2009 The MathWorks, Inc.

    properties (SetAccess=private,GetAccess=private)
        aser   % Serial Connection
    end

    methods

        % constructor, connects to the board and creates an arduino object
        function a=arduino(comPort)

            % Add target directories and save the updated path
            addpath(fullfile(pwd));
            savepath

            % check nargin
            if nargin<1,
                comPort='DEMO';
                disp('Note: a DEMO connection will be created');
                disp('Use a the com port, e.g. "COM5" as input argument to connect
to the real board');
            end

            % check port
            if ~ischar(comPort),
                error('The input argument must be a string, e.g. "COM8" ');
            end

            % check if we are already connected
            if isa(a.aser,'serial') && isvalid(a.aser) &&
strcmpi(get(a.aser,'Status'),'open'),
                disp(['It looks like Arduino is already connected to port ' comPort ]);
                disp('Delete the object to force disconnection');
                disp('before attempting a connection to a different port.');
```

```
    return;
end

% check whether serial port is currently used by MATLAB
if ~isempty(instrfind({'Port'},{comPort})),
    disp(['The port ' comPort ' is already used by MATLAB']);
    disp(['If you are sure that Arduino is connected to ' comPort]);
    disp('then delete the object to disconnect and execute:');
    disp(['  delete(instrfind({''Port''},{''' comPort '''}))']);
    disp('to delete the port before attempting another connection');
    error(['Port ' comPort ' already used by MATLAB']);
end

% define serial object
a.aser=serial(comPort);

% connection
if strcmpi(get(a.aser,'Port'),'DEMO'),
    % handle demo mode

    fprintf(1,'Demo mode connection ..');
    for i=1:4,
        fprintf(1,'.');
        pause(1);
    end
    fprintf(1,'\n');
    pause(1);

    % chk is 1 or 2 depending on the script running on the board
    chk=round(1+rand);

else
    % actual connection

    % open port
    try
        fopen(a.aser);
    catch ME,
        disp(ME.message)
        delete(a);
        error(['Could not open port: ' comPort]);
    end

    % it takes several seconds before any operation could be attempted
```

```
            fprintf(1,'Attempting connection ..');
            for i=1:4,
                fprintf(1,'.');
                pause(1);
            end
            fprintf(1,'\n');

            % query script type
            fwrite(a.aser,[57 57],'uchar');
            chk=fscanf(a.aser,'%d');

            % exit if there was no answer
            if isempty(chk)
                delete(a);
                error('Connection unsuccessful, please make sure that the Arduino
is powered on, running either adiosrv.pde or mororsrv.pde, and that the board is
connected to the indicated serial port. You might also try to unplug and re-plug
the USB cable before attempting a reconnection.');
            end

        end

        % check returned value
        if chk==1,
            disp('Basic I/O Script detected !');
        elseif chk==2,
            disp('Motor Shield Script detected !');
        else
            delete(a);
            error('Unknown Script. Please make sure that either adiosrv.pde or
motorsrv.pde are running on the Arduino');
        end

        % sets a.mots flag
        a.mots=chk-1;

        % set a.aser tag
        a.aser.Tag='ok';

        % initialize pin vector (-1 is unassigned, 0 is input, 1 is output)
        a.pins=-1*ones(1,19);

        % initialize servo vector (-1 is unknown, 0 is detached, 1 is attached)
```

```
            a.srvs=0*ones(1,2);

            % initialize motor vector (0 to 255 is the speed)
            a.mspd=0*ones(1,4);

            % initialize stepper vector (0 to 255 is the speed)
            a.sspd=0*ones(1,2);

            % notify successful installation
            disp('Arduino successfully connected !');

        end % arduino

        % distructor, deletes the object
        function delete(a)

            % if it is a serial, valid and open then close it
            if isa(a.aser,'serial') && isvalid(a.aser) &&
strcmpi(get(a.aser,'Status'),'open'),
                if ~isempty(a.aser.Tag),
                    try
                        % trying to leave it in a known unharmful state
                        for i=2:19,
                            a.pinMode(i,'output');
                            a.digitalWrite(i,0);
                            a.pinMode(i,'input');
                        end
                    catch ME
                        % disp but proceed anyway
                        disp(ME.message);
                        disp('Proceeding to deletion anyway');
                    end

                end
                fclose(a.aser);
            end

            % if it's an object delete it
            if isobject(a.aser),
                delete(a.aser);
            end


        end % delete
```

```
        % disp, displays the object
        function disp(a) % display
            if isvalid(a),
                if isa(a.aser,'serial') && isvalid(a.aser),
                    disp(['<a href="matlab:help arduino">arduino</a> object
connected to ' a.aser.port ' port']);
                    if a.mots==1,
                        disp('Motor Shield Server running on the arduino board');
                        disp(' ');
                        a.servoStatus
                        a.motorSpeed
                        a.stepperSpeed
                        disp(' ');
                        disp('Servo Methods: <a href="matlab:help
servoStatus">servoStatus</a> <a href="matlab:help
servoAttach">servoAttach</a> <a href="matlab:help
servoDetach">servoDetach</a> <a href="matlab:help
servoRead">servoRead</a> <a href="matlab:help
servoWrite">servoWrite</a>');
                        disp('DC Motors and Stepper Methods: <a href="matlab:help
motorSpeed">motorSpeed</a> <a href="matlab:help
motorRun">motorRun</a> <a href="matlab:help
stepperSpeed">stepperSpeed</a> <a href="matlab:help
stepperStep">stepperStep</a>');
                    else
                        disp('IO Server running on the arduino board');
                        disp(' ');
                        a.pinMode
                        disp(' ');
                        disp('Pin IO Methods: <a href="matlab:help
pinMode">pinMode</a> <a href="matlab:help digitalRead">digitalRead</a>
<a href="matlab:help digitalWrite">digitalWrite</a> <a href="matlab:help
analogRead">analogRead</a> <a href="matlab:help
analogWrite">analogWrite</a>');
                    end
                    disp(' ');
                else
                    disp('<a href="matlab:help arduino">arduino</a> object
connected to an invalid serial port');
                    disp('Please delete the arduino object');
                    disp(' ');
                end
            else
```

```
            disp('Invalid <a href="matlab:help arduino">arduino</a> object');
            disp('Please clear the object and instantiate another one');
            disp(' ');
        end
    end

    % pin mode, changes pin mode
    function pinMode(a,pin,str)

        % a.pinMode(pin,str); specifies the pin mode of a digital pins.
        % The first argument before the function name, a, is the arduino object.
        % The first argument, pin, is the number of the digital pin (2 to 19).
        % The second argument, str, is a string that can be 'input' or 'output',
        % Called with one argument, as a.pin(pin) it returns the mode of
        % the digital pin, called without arguments, prints the mode of all the
        % digital pins. Note that the digital pins from 0 to 13 are located on
        % the upper right part of the board, while the digital pins from 14 to 19
        % are better known as "analog input" pins and are located in the lower
        % right corner of the board.
        %
        % Examples:
        % a.pinMode(11,'output') % sets digital pin #11 as output
        % a.pinMode(10,'input')  % sets digital pin #10 as input
        % val=a.pinMode(10);     % returns the status of digital pin #10
        % a.pinMode(5);          % prints the status of digital pin #5
        % a.pinMode;             % prints the status of all pins
        %


        %%%%%%%%%%%%%%%%%%%%%%%% ARGUMENT
CHECKING %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        % check nargin
        if nargin>3,
            error('This function cannot have more than 3 arguments, object, pin
and str');
        end

        % first argument must be the arduino variable
        if ~isa(a,'arduino'), error('The first argument must be an arduino
variable'); end

        % if pin argument is there check it
        if nargin>1,
```

```
    errstr=arduino.checknum(pin,'pin number',2:19);
    if ~isempty(errstr), error(errstr); end
end

% if str argument is there check it
if nargin>2,
    errstr=arduino.checkstr(str,'pin mode',{'input','output'});
    if ~isempty(errstr), error(errstr); end
end

% perform the requested action
if nargin==3,

    % check a.aser for validity
    errstr=arduino.checkser(a.aser,'valid');
    if ~isempty(errstr), error(errstr); end

    %%%%%%%%%%%%%%%%%%%%%%%%%% CHANGE PIN
MODE %%%%%%%%%%%%%%%%%%%%
```

```
            % store 0 for input and 1 for output
            a.pins(pin)=val;

        elseif nargin==2,
            % print pin mode for the requested pin

            mode={'UNASSIGNED','set as INPUT','set as OUTPUT'};
            disp(['Digital Pin ' num2str(pin) ' is currently ' mode{2+a.pins(pin)}]);

        else
            % print pin mode for each pin

            mode={'UNASSIGNED','set as INPUT','set as OUTPUT'};
            for i=2:19;
                disp(['Digital Pin ' num2str(i,'%02d') ' is currently '
mode{2+a.pins(i)}]);
            end

        end

    end % pinmode

    % digital read
    function val=digitalRead(a,pin)

        % val=a.digitalRead(pin); performs digital input on a given arduino pin.
        % The first argument before the function name, a, is the arduino object.
        % The argument pin, is the number of the digital pin (2 to 19)
        % where the digital input needs to be performed. Note that the digital
pins
        % from 0 to 13 are located on the upper right part of the board, while
the
        % digital pins from 14 to 19 are better known as "analog input" pins and
        % are located in the lower right corner of the board.
        %
        % Example:
        % val=a.digitalRead(4); % reads pin #4
        %


        %%%%%%%%%%%%%%%%%%%%%%%%%% ARGUMENT
CHECKING %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
        % check nargin
        if nargin~=2,
            error('Function must have the "pin" argument');
        end

        % first argument must be the arduino variable
        if ~isa(a,'arduino'), error('The first argument must be an arduino
variable'); end

        % check pin
        errstr=arduino.checknum(pin,'pin number',2:19);
        if ~isempty(errstr), error(errstr); end

        % check a.aser for validity
        errstr=arduino.checkser(a.aser,'valid');
        if ~isempty(errstr), error(errstr); end


        %%%%%%%%%%%%%%%%%%%%%%%%% PERFORM
DIGITAL INPUT %%%%%%%%%%%%%%%%%%%%%%%%%%%%

        if strcmpi(get(a.aser,'Port'),'DEMO'),
            % handle demo mode

            % average digital input delay
            pause(0.0247);

            % output 0 or 1 randomly
            val=round(rand);

        else

            % check a.aser for openness
            errstr=arduino.checkser(a.aser,'open');
            if ~isempty(errstr), error(errstr); end

            % send mode and pin
            fwrite(a.aser,[49 97+pin],'uchar');

            % get value
            val=fscanf(a.aser,'%d');

        end
```

```
    end % digitalread

    % digital write
    function digitalWrite(a,pin,val)

        % a.digitalWrite(pin,val); performs digital output on a given pin.
        % The first argument before the function name, a, is the arduino object.
        % The second argument, pin, is the number of the digital pin (2 to 19)
        % where the digital output needs to be performed.
        % The third argument, val, is the value (either 0 or 1) for the output
        % Note that the digital pins from 0 to 13 are located on the upper right
part
        % of the board, while the digital pins from 14 to 19 are better known as
        % "analog input" pins and are located in the lower right corner of the
board.
        %
        % Examples:
        % a.digitalWrite(13,1); % sets pin #13 high
        % a.digitalWrite(13,0); % sets pin #13 low
        %


        %%%%%%%%%%%%%%%%%%%%%%%%%% ARGUMENT
CHECKING %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        % check nargin
        if nargin~=3,
           error('Function must have the "pin" and "val" arguments');
        end

        % first argument must be the arduino variable
        if ~isa(a,'arduino'), error('The first argument must be an arduino
variable'); end

        % check pin
        errstr=arduino.checknum(pin,'pin number',2:19);
        if ~isempty(errstr), error(errstr); end

        % check val
        errstr=arduino.checknum(val,'value',0:1);
        if ~isempty(errstr), error(errstr); end

        % pin should be configured as output
        if a.pins(pin)~=1,
```

```
            warning('MATLAB:Arduino:digitalWrite',['If digital pin ' num2str(pin) '
is set as input, digital output takes place only after using a.pinMode('
num2str(pin) ',"output"); ']);
        end

        % check a.aser for validity
        errstr=arduino.checkser(a.aser,'valid');
        if ~isempty(errstr), error(errstr); end


        %%%%%%%%%%%%%%%%%%%%%%%%% PERFORM
DIGITAL OUTPUT %%%%%%%%%%%%%%%%%%%%%%%%%%%%

        if strcmpi(get(a.aser,'Port'),'DEMO'),
            % handle demo mode

            % average digital output delay
            pause(0.0087);

        else

            % check a.aser for openness
            errstr=arduino.checkser(a.aser,'open');
            if ~isempty(errstr), error(errstr); end

            % send mode, pin and value
            fwrite(a.aser,[50 97+pin 48+val],'uchar');

        end

    end % digitalwrite

    % analog read
    function val=analogRead(a,pin)

        % val=a.analogRead(pin); Performs analog input on a given arduino
pin.
        % The first argument before the function name, a, is the arduino object.
        % The second argument, pin, is the number of the analog input pin (0
to 5)
        % where the analog input needs to be performed. The returned value,
val,
        % ranges from 0 to 1023, with 0 corresponding to an input voltage of 0
volts,
```

```
        % and 1023 to a value of 5 volts. Therefore the resolution is .0049 volts
        % (4.9 mV) per unit.
        % Note that the analog input pins 0 to 5 are also known as digital pins
        % from 14 to 19, and are located on the lower right corner of the board.
        % Specifically, analog input pin 0 corresponds to digital pin 14, and
analog
        % input pin 5 corresponds to digital pin 19. Performing analog input
does
        % not affect the digital state (high, low, digital input) of the pin.
        %
        % Example:
        % val=a.analogRead(0); % reads analog input pin # 0
        %


        %%%%%%%%%%%%%%%%%%%%%%%%%% ARGUMENT
CHECKING %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        % check nargin
        if nargin~=2,
            error('Function must have the "pin" argument');
        end

        % first argument must be the arduino variable
        if ~isa(a,'arduino'), error('The first argument must be an arduino
variable'); end

        % check pin
        errstr=arduino.checknum(pin,'analog input pin number',0:5);
        if ~isempty(errstr), error(errstr); end

        % check a.aser for validity
        errstr=arduino.checkser(a.aser,'valid');
        if ~isempty(errstr), error(errstr); end

        %%%%%%%%%%%%%%%%%%%%%%%%%% PERFORM
ANALOG INPUT %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        if strcmpi(get(a.aser,'Port'),'DEMO'),
            % handle demo mode

            % average analog input delay
            pause(0.0267);
```

```
            % output a random value between 0 and 1023
            val=round(1023*rand);

        else

            % check a.aser for openness
            errstr=arduino.checkser(a.aser,'open');
            if ~isempty(errstr), error(errstr); end

            % send mode and pin
            fwrite(a.aser,[51 97+pin],'uchar');

            % get value
            val=fscanf(a.aser,'%d');

        end

    end % analogread

    % function analog write
    function analogWrite(a,pin,val)

        % a.analogWrite(pin,val); Performs analog output on a given arduino
pin.
        % The first argument before the function name, a, is the arduino object.
        % The first argument, pin, is the number of the DIGITAL pin where the
analog
        % (PWM) output needs to be performed. Allowed pins for AO are
3,5,6,9,10,11
        % The second argument, val, is the value from 0 to 255 for the level of
        % analog output. Note that the digital pins from 0 to 13 are located on
the
        % upper right part of the board.
        %
        % Examples:
        % a.analogWrite(11,90); % sets pin #11 to 90/255
        % a.analogWrite(3,10); % sets pin #3 to 10/255
        %

        %%%%%%%%%%%%%%%%%%%%%%%%%% ARGUMENT
CHECKING %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

        % check nargin
        if nargin~=3,
```

```
            error('Function must have the "pin" and "val" arguments');
        end

        % first argument must be the arduino variable
        if ~isa(a,'arduino'), error('The first argument must be an arduino
variable'); end

        % check pin
        errstr=arduino.checknum(pin,'pwm pin number',[3 5 6 9 10 11]);
        if ~isempty(errstr), error(errstr); end

        % check val
        errstr=arduino.checknum(val,'analog output level',0:255);
        if ~isempty(errstr), error(errstr); end

        % pin should be configured as output
        if a.pins(pin)~=1,
            warning('MATLAB:Arduino:analogWrite',['If digital pin ' num2str(pin) '
is set as input, pwm output takes place only after using a.pinMode('
num2str(pin) ',''output''); ']);
        end

        % check a.aser for validity
        errstr=arduino.checkser(a.aser,'valid');
        if ~isempty(errstr), error(errstr); end


        %%%%%%%%%%%%%%%%%%%%%%%% PERFORM
ANALOG OUTPUT %%%%%%%%%%%%%%%%%%%%%%%%%%%%

        if strcmpi(get(a.aser,'Port'),'DEMO'),
            % handle demo mode

            % average analog output delay
            pause(0.0088);

        else

            % check a.aser for openness
            errstr=arduino.checkser(a.aser,'open');
            if ~isempty(errstr), error(errstr); end

            % send mode, pin and value
            fwrite(a.aser,[52 97+pin val],'uchar');
```

```
        end


    end % analogwrite


end % methods

methods (Static) % static methods

    function errstr=checknum(num,description,allowed)

        % errstr=arduino.checknum(num,description,allowed); Checks numeric
argument.
        % This function checks the first argument, num, described in the string
        % given as a second argument, to make sure that it is real, scalar,
        % and that it is equal to one of the entries of the vector of allowed
        % values given as a third argument. If the check is successful then the
        % returned argument is empty, otherwise it is a string specifying
        % the type of error.

        % preliminary: check nargin
        if nargin~=3,
            error('checknum needs 3 arguments, please read the help');
        end

        % preliminary: check description
        if isempty(description) || ~ischar(description)
            error('checknum second argument must be a string');
        end

        % preliminary: check allowed
        if isempty(allowed) || ~isnumeric(allowed)
            error('checknum third argument must be a numeric vector');
        end

        % initialize error string
        errstr=[];

        % check num for type
        if ~isnumeric(num),
            errstr=['The ' description ' must be numeric'];
            return
```

```
            end

            % check num for size
            if numel(num)~=1,
               errstr=['The ' description ' must be a scalar'];
               return
            end

            % check num for realness
            if ~isreal(num),
               errstr=['The ' description ' must be a real value'];
               return
            end

            % check num against allowed values
            if ~any(allowed==num),

               % form right error string
               if numel(allowed)==1,
                  errstr=['Unallowed value for ' description ', the value must be
exactly ' num2str(allowed(1))];
               elseif numel(allowed)==2,
                  errstr=['Unallowed value for ' description ', the value must be
either ' num2str(allowed(1)) ' or ' num2str(allowed(2))];
               elseif max(diff(allowed))==1,
                  errstr=['Unallowed value for ' description ', the value must be an
integer going from ' num2str(allowed(1)) ' to ' num2str(allowed(end))];
               else
                  errstr=['Unallowed value for ' description ', the value must be one
of the following: ' mat2str(allowed)];
               end

            end

         end % checknum

         function errstr=checkstr(str,description,allowed)

            % errstr=arduino.checkstr(str,description,allowed); Checks string
argument.
            % This function checks the first argument, str, described in the string
            % given as a second argument, to make sure that it is a string, and that
            % its first character is equal to one of the entries in the cell of
```

```
        % allowed characters given as a third argument. If the check is
successful
        % then the returned argument is empty, otherwise it is a string
specifying
        % the type of error.

        % preliminary: check nargin
        if nargin~=3,
           error('checkstr needs 3 arguments, please read the help');
        end

        % preliminary: check description
        if isempty(description) || ~ischar(description)
           error('checknum second argument must be a string');
        end

        % preliminary: check allowed
        if ~iscell(allowed) || numel(allowed)<2,
           error('checknum third argument must be a cell with at least 2
entries');
        end

        % initialize error string
        errstr=[];

        % check string for type
        if ~ischar(str),
           errstr=['The ' description ' argument must be a string'];
           return
        end

        % check string for size
        if numel(str)<1,
           errstr=['The ' description ' argument cannot be empty'];
           return
        end

        % check str against allowed values
        if ~any(strcmpi(str,allowed)),

           % make sure this is a hozizontal vector
           allowed=allowed(:)';

           % add a comma at the end of each value
```

```
            for i=1:length(allowed)-1,
                allowed{i}=['"' allowed{i} '", '];
            end

            % form error string
            errstr=['Unallowed value for ' description ', the value must be either: '
allowed{1:end-1} 'or "' allowed{end} '"'];
            return
        end

    end % checkstr

    function errstr=checkser(ser,chk)

        % errstr=arduino.checkser(ser,chk); Checks serial connection argument.
        % This function checks the first argument, ser, to make sure that either:
        % 1) it is a valid serial connection (if the second argument is 'valid')
        % 3) it is open (if the second argument is 'open')
        % If the check is successful then the returned argument is empty,
        % otherwise it is a string specifying the type of error.

        % preliminary: check nargin
        if nargin~=2,
            error('checkser needs two arguments, please read the help');
        end

        % initialize error string
        errstr=[];

        % check serial connection
        switch lower(chk),

            case 'valid',

                % make sure is a serial port
                if ~isa(ser,'serial'),
                    disp('Arduino is not connected, please re-create the object
before using this function.');
                    errstr='Arduino not connected';
                    return
                end

                % make sure is valid
                if ~isvalid(ser),
```

```
                    disp('Serial connection invalid, please recreate the object to
reconnect to a serial port.');
                    errstr='Serial connection invalid';
                    return
                end

            case 'open',

                % check openness
                if ~strcmpi(get(ser,'Status'),'open'),
                    disp('Serial connection not opened, please recreate the object to
reconnect to a serial port.');
                    errstr='Serial connection not opened';
                    return
                end


            otherwise

                % complain
                error('second argument must be either "valid" or "open"');

        end

    end % chackser

end % static methods

end % class def
```

◆◆

irisrecognition.m

```
function varargout = irisrecognition(varargin)

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @irisrecognition_OpeningFcn, ...
                   'gui_OutputFcn',  @irisrecognition_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
```

```
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before untitled is made visible.
function irisrecognition_OpeningFcn(hObject, eventdata, handles, varargin)

%setup webcam
%vidobj = videoinput('winvideo',1,'YUY2_640X480');
%set(handles.statusLbl,'String','Connecting to camera');
vidobj = videoinput('winvideo',3,'RGB24_640X480');
axes(handles.CameraAxes);
videoRes = get(vidobj, 'VideoResolution');
numberOfBands = get(vidobj, 'NumberOfBands');
fprintf(1, 'Video resolution = %d wide by %d tall, by %d color',...
    videoRes(1), videoRes(2), numberOfBands);
handleToImage = image( zeros([videoRes(2), videoRes(1),...
numberOfBands], 'uint8') );
set(vidobj,'ReturnedColorSpace','RGB');
uint8('img');
%set(handles.statusLbl,'String','Camera ready!');
preview(vidobj, handleToImage);
%end setup webcam

%set arduino
%set(handles.statusLbl,'String','Connecting to Gizduino');
gizduinoPort = arduino('COM4');
%set(handles.statusLbl,'String','Gizduino ready!');
gizduinoPort.pinMode(8, 'input');%where the output of the proximity sensor goes
gizduinoPort.pinMode(11, 'output');% made a +5V supply on pin 11 for the +V
of the proximity sensor
gizduinoPort.digitalWrite(11, 1);%output high signal on pin 11 (5V)
%end set arduino
while gizduinoPort ~= 0     %habang nakaconnect ang microcontroller
      myWait(5);%wait for (n) seconds
      if gizduinoPort.digitalRead(8) == 0  %kapag may harang
     %       set(handles.statusLbl,'String','Capturing iris image . . .');
```

```
        frame = getsnapshot(vidobj); %capture
        imwrite(frame,'F:\tempimage.bmp'); %save
   %       set(handles.statusLbl,'String','Saving . . .');
        imwrite(frame,'F:\Pictures\tempimage.bmp');
        imwrite(frame,'F:\Pictures\tempimage1.bmp');
        %match
  %       set(handles.statusLbl,'String','Searching for matches . . .');
        output = irisrecognitionprocess('F:\tempimage.bmp');
 %        set(handles.statusLbl,'String','Searching Completed!');
%         frmAddName
        conn = database('thesis','sa','mssql');
        cursor = exec(conn,'select IrisId,IrisTemplate from
IrisDataBankDesign');
        cursor = fetch(cursor);
        intmax = size(cursor.data,1);
        mat1 = output;
        output = mat2str(mat1);
        mat1 = str2mat(output);
        irisfound = 0;
        threshold = 0.1000;
        int = 0;
        for int = 1:intmax
            mat2 = str2mat(cursor.data(int,2));
            HD = gethammingdistance(mat1,mat2);
            HD_values(int) = HD;
            statuslBl = HD;
            if HD > 0 & HD < threshold
                irisfound = 1;
            end
        end
        set(handles.HdListbox,'String',HD_values);
        close(cursor);
        close(conn);
        if irisfound == 1
            msgbox('Authenticated!','IRIS RECOGNITION')
        else
            msgbox('Iris Not Authenticated!','IRIS RECOGNITION')
            myWait(2);
            set(handles.EnrollBtn,'Enable','on');
        end
         %end match
    else
%         set(handles.statusLbl,'String','Idle');
    end
```

```
end

% Choose default command line output for untitled
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes untitled wait for user response (see UIRESUME)
% uiwait(handles.figure1);
% --- Outputs from this function are returned to the command line.
function varargout = irisrecognition_OutputFcn(hObject, eventdata, handles)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in exitBtn.
function exitBtn_Callback(hObject, eventdata, handles)

close(handles.figure1)


% --- Executes on selection change in HdListbox.
function HdListbox_Callback(hObject, eventdata, handles)

Hints: contents = get(hObject,'String') returns HdListbox contents as cell array
%        contents{get(hObject,'Value')} returns selected item from HdListbox


% --- Executes during object creation, after setting all properties.
function HdListbox_CreateFcn(hObject, eventdata, handles)

if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in EnrollBtn.
function EnrollBtn_Callback(hObject, eventdata, handles)
frmAddName
```

◆◆◆

```
% Write noise regions
imagewithnoise2(ind2) = 255;
imagewithnoise2(ind1) = 255;

% Write circles overlayed
imagewithcircles(ind2) = 255;
imagewithcircles(ind1) = 255;
w = cd;
cd(DIAGPATH);

imwrite(imagewithcircles,[eyeimage_filename,'-segmented.jpg'],'jpg');
cd(w);

% perform normalisation

[polar_array noise_array] = normaliseiris(imagewithnoise, circleiris(2),...
   circleiris(1), circleiris(3), circlepupil(2), circlepupil(1),
circlepupil(3),eyeimage_filename, radial_res, angular_res);


% WRITE NORMALISED PATTERN, AND NOISE PATTERN
w = cd;
cd(DIAGPATH);
imwrite(polar_array,[eyeimage_filename,'-polar.jpg'],'jpg');
cd(w);

%ENCODE THE TEMPLATE USING WAVELET
%[output] = encode(polar_array, noise_array, nscales, minWaveLength, mult,
%sigmaOnf);
[output] = encode(polar_array);
```

◆◆◆

◆◆◆

```
position(2) = (screenSize(4)-position(4))/2;

%center the window
set( handles.frmAddName, ...
    'Position', position );


% --- Outputs from this function are returned to the command line.
function varargout = frmAddName_OutputFcn(hObject, eventdata, handles)

% Get default command line output from handles structure
varargout{1} = handles.output;


function txtName_Callback(hObject, eventdata, handles)

% --- Executes during object creation, after setting all properties.
function txtName_CreateFcn(hObject, eventdata, handles)
if ispc
    set(hObject,'BackgroundColor','white');
else
    set(hObject,'BackgroundColor',get(0,'defaultUicontrolBackgroundColor'));
end


% --- Executes on button press in btnOK.
function btnOK_Callback(hObject, eventdata, handles)
frame = imread('F:\Pictures\tempimage.bmp');

HH = findobj(gcf,'Tag','txtName');
ID = get(HH,'String');
IDcat = strcat('F:\Pictures\',ID);
filename = strcat(IDcat,'.jpg');
if exist(filename,'file')
errordlg('Name already exist.','Information');
return
else

imwrite(frame,filename,'JPG');
output = irisrecognitionprocess(filename);
conn = database('thesis','sa','mssql');
colnames = {'IrisPath','IrisTemplate'};
output = mat2str(output);
exdata = {filename,output};
```

```
insert(conn,'iris.dbo.IrisDataBankDesign', colnames, exdata)
end
close(conn);
close;

% --- Executes on button press in btnCancel.
function btnCancel_Callback(hObject, eventdata, handles)
close;
```

◆◆

segmentiris.m

```
% segmentiris - peforms automatic segmentation of the iris region
% from an eye image. Also isolates noise areas such as occluding
% eyelids and eyelashes.
%
% Usage:
% [circleiris, circlepupil, imagewithnoise] = segmentiris(image)
%
% Arguments:
%   eyeimage        - the input eye image
%
% Output:
%   circleiris      - centre coordinates and radius
%                     of the detected iris boundary
%   circlepupil     - centre coordinates and radius
%                     of the detected pupil boundary
%   imagewithnoise  - original eye image, but with
%                     location of noise marked with
%                     NaN values

function [circleiris, circlepupil, imagewithnoise] = segmentiris(eyeimage)
lpupilradius = 20;
upupilradius = 75;
lirisradius = 80;
uirisradius = 95;

% define scaling factor to speed up Hough transform
scaling = 0.4;

reflecthres = 240;

% find the iris boundary
```

```
[row, col, r] = findcircle(eyeimage, lirisradius, uirisradius, scaling, 2, 0.20, 0.19,
1.00, 0.00);

circleiris = [row col r];

rowd = double(row);
cold = double(col);
rd = double(r);

irl = round(rowd-rd);
iru = round(rowd+rd);
icl = round(cold-rd);
icu = round(cold+rd);

imgsize = size(eyeimage);

if irl < 1
   irl = 1;
end

if icl < 1
   icl = 1;
end

if iru > imgsize(1)
   iru = imgsize(1);
end

if icu > imgsize(2)
   icu = imgsize(2);
end

% to find the inner pupil, use just the region within the previously
% detected iris boundary
imagepupil = eyeimage( irl:iru,icl:icu);

%find pupil boundary
[rowp, colp, r] = findcircle(imagepupil, lpupilradius, upupilradius
,0.6,2,0.25,0.25,1.00,1.00);

rowp = double(rowp);
colp = double(colp);
r = double(r);
```

```
row = double(irl) + rowp;
col = double(icl) + colp;

row = round(row);
col = round(col);

circlepupil = [row col r];

% set up array for recording noise regions
% noise pixels will have NaN values
imagewithnoise = double(eyeimage);

%find top eyelid
topeyelid = imagepupil(1:(rowp-r),:);
lines = findline(topeyelid);

if size(lines,1) > 0
    [xl yl] = linecoords(lines, size(topeyelid));
    yl = double(yl) + irl-1;
    xl = double(xl) + icl-1;

    yla = max(yl);

    y2 = 1:yla;

    ind3 = sub2ind(size(eyeimage),yl,xl);
    imagewithnoise(ind3) = NaN;

    imagewithnoise(y2, xl) = NaN;
end

%find bottom eyelid
bottomeyelid = imagepupil((rowp+r):size(imagepupil,1),:);
lines = findline(bottomeyelid);

if size(lines,1) > 0

    [xl yl] = linecoords(lines, size(bottomeyelid));
    yl = double(yl)+ irl+rowp+r-2;
    xl = double(xl) + icl-1;

    yla = min(yl);

    y2 = yla:size(eyeimage,1);
```

```
    ind4 = sub2ind(size(eyeimage),yl,xl);
    imagewithnoise(ind4) = NaN;
    imagewithnoise(y2, xl) = NaN;

end

ref = eyeimage < 100;
coords = find(ref==1);
imagewithnoise(coords) = NaN;
```

◆◆◆

```
if radius < 1
  error('radius must be >= 1');
end

[rows,cols] = size(inimage);
im = zeros(rows,cols);        % Preallocate memory for output image for speed
iradius = ceil(radius);

% Precalculate x and y offsets relative to centre pixel for each orientation angle

angle = [0:180].*pi/180;    % Array of angles in 1 degree increments (but in
radians).
xoff = radius*cos(angle);   % x and y offset of points at specified radius and
angle
yoff = radius*sin(angle);   % from each reference position.

hfrac = xoff - floor(xoff); % Fractional offset of xoff relative to integer location
vfrac = yoff - floor(yoff); % Fractional offset of yoff relative to integer location

orient = fix(orient)+1;     % Orientations start at 0 degrees but arrays start
                            % with index 1.

% Now run through the image interpolating grey values on each side
% of the centre pixel to be used for the non-maximal suppression.

for row = (iradius+1):(rows - iradius)
  for col = (iradius+1):(cols - iradius)

    or = orient(row,col);   % Index into precomputed arrays

    x = col + xoff(or);     % x, y location on one side of the point in question
    y = row - yoff(or);

    fx = floor(x);          % Get integer pixel locations that surround location x,y
    cx = ceil(x);
    fy = floor(y);
    cy = ceil(y);
    tl = inimage(fy,fx);    % Value at top left integer pixel location.
    tr = inimage(fy,cx);    % top right
    bl = inimage(cy,fx);    % bottom left
    br = inimage(cy,cx);    % bottom right

    upperavg = tl + hfrac(or) * (tr - tl);  % Now use bilinear interpolation to
    loweravg = bl + hfrac(or) * (br - bl);  % estimate value at x,y
```

```
    v1 = upperavg + vfrac(or) * (loweravg - upperavg);

  if inimage(row, col) > v1 % We need to check the value on the other side...

    x = col - xoff(or);      % x, y location on the `other side' of the point in
question
    y = row + yoff(or);

    fx = floor(x);
    cx = ceil(x);
    fy = floor(y);
    cy = ceil(y);
    tl = inimage(fy,fx);     % Value at top left integer pixel location.
    tr = inimage(fy,cx);     % top right
    bl = inimage(cy,fx);     % bottom left
    br = inimage(cy,cx);     % bottom right
    upperavg = tl + hfrac(or) * (tr - tl);
    loweravg = bl + hfrac(or) * (br - bl);
    v2 = upperavg + vfrac(or) * (loweravg - upperavg);

    if inimage(row,col) > v2            % This is a local maximum.
      im(row, col) = inimage(row, col); % Record value in the output image.
    end

   end
  end
end
```

◆◆◆

```
%   y          - corresponding y coordinates
%

function [x,y] = linecoords(lines, imsize)

xd = [1:imsize(2)];
yd = (-lines(3) - lines(1)*xd ) / lines(2);

coords = find(yd>imsize(1));
yd(coords) = imsize(1);
coords = find(yd<1);
yd(coords) = 1;

x = int32(xd);
y = int32(yd);
```

◆◆

hysthresh.m

```
% HYSTHRESH - Hysteresis thresholding
%
% Usage: bw = hysthresh(im, T1, T2)
%
% Arguments:
%           im  - image to be thresholded (assumed to be non-negative)
%           T1  - upper threshold value
%           T2  - lower threshold value
%
% Returns:
%           bw  - the thresholded image (containing values 0 or 1)
%
% Function performs hysteresis thresholding of an image.
% All pixels with values above threshold T1 are marked as edges
% All pixels that are adjacent to points that have been marked as edges
% and with values above threshold T2 are also marked as edges. Eight
% connectivity is used.
%
% It is assumed that the input image is non-negative
%

function bw = hysthresh(im, T1, T2)

if (T2 > T1 | T2 < 0 | T1 < 0)  % Check thesholds are sensible
  error('T1 must be >= T2 and both must be >= 0 ');
```

```
end

[rows, cols] = size(im);    % Precompute some values for speed and
convenience.
rc = rows*cols;
rcmr = rc - rows;
rp1 = rows+1;

bw = im(:);                 % Make image into a column vector
pix = find(bw > T1);        % Find indices of all pixels with value > T1
npix = size(pix,1);         % Find the number of pixels with value > T1

stack = zeros(rows*cols,1); % Create a stack array (that should never
                            % overflow!)

stack(1:npix) = pix;        % Put all the edge points on the stack
stp = npix;                 % set stack pointer
for k = 1:npix
   bw(pix(k)) = -1;         % mark points as edges
end


% Precompute an array, O, of index offset values that correspond to the eight
% surrounding pixels of any point. Note that the image was transformed into
% a column vector, so if we reshape the image back to a square the indices
% surrounding a pixel with index, n, will be:
%              n-rows-1   n-1   n+rows-1
%
%               n-rows     n     n+rows
%
%              n-rows+1   n+1   n+rows+1

O = [-1, 1, -rows-1, -rows, -rows+1, rows-1, rows, rows+1];

while stp ~= 0              % While the stack is not empty
   v = stack(stp);          % Pop next index off the stack
   stp = stp - 1;

   if v > rp1 & v < rcmr    % Prevent us from generating illegal indices
                % Now look at surrounding pixels to see if they
                % should be pushed onto the stack to be
                % processed as well.
      index = O+v;      % Calculate indices of points around this pixel.
      for l = 1:8
```

```
      ind = index(l);
      if bw(ind) > T2   % if value > T2,
         stp = stp+1;  % push index onto the stack.
         stack(stp) = ind;
         bw(ind) = -1; % mark this as an edge point
      end
      end
   end
end



bw = (bw == -1);            % Finally zero out anything that was not an edge
bw = reshape(bw,rows,cols); % and reshape the image
```

◆◆◆

```
    cy = y(index);

    for n=1:nradii

        h(:,:,n) = addcircle(h(:,:,n),[cx,cy],n+rmin);

    end

end
```

◆◆◆

```
    lines = [];
    return;
end

[foo, ind] = sort(-R(i));
u = size(i,1);
k = i(ind(1:u));
[y,x]=ind2sub(size(R),k);
t = -theta(x)*pi/180;
r = xp(y);

lines = [cos(t) sin(t) -r];

cx = size(image,2)/2-1;
cy = size(image,1)/2-1;
lines(:,3) = lines(:,3) - lines(:,1)*cx - lines(:,2)*cy;
```

◆◆◆

```
%   circlepupil     - centre coordinates and radius
%                   of the detected pupil boundary
%   imagewithnoise  - original eye image, but with
%                   location of noise marked with
%                   NaN values
%

function [row, col, r] = findcircle(image,lradius,uradius,scaling, sigma, hithres,
lowthres, vert, horz)

lradsc = round(lradius*scaling);
uradsc = round(uradius*scaling);
rd = round(uradius*scaling - lradius*scaling);

% generate the edge image
[I2 or] = canny(image, sigma, scaling, vert, horz);
%1.9 to 1.5 for gamma
I3 = adjgamma(I2, 1.8);
I4 = nonmaxsup(I3, or, 1.5);
edgeimage = hysthresh(I4, hithres, lowthres);

% perform the circular Hough transform
h = houghcircle(edgeimage, lradsc, uradsc);

maxtotal = 0;

% find the maximum in the Hough space, and hence
% the parameters of the circle
for i=1:rd

    layer = h(:,:,i);
    [maxlayer] = max(max(layer));


    if maxlayer > maxtotal

        maxtotal = maxlayer;


        r = int32((lradsc+i) / scaling);

        [row,col] = ( find(layer == maxlayer) );
```

```
        row = int32(row(1) / scaling); % returns only first max value
        col = int32(col(1) / scaling);

    end

end
```

◆◆◆

```
lradsc = round(lradius*scaling);
uradsc = round(uradius*scaling);
rd = round(uradius*scaling - lradius*scaling);

% generate the edge image
[I2 or] = canny(image, sigma, scaling, vert, horz);
%1.9 to 1.5 for gamma
I3 = adjgamma(I2, 1.8);
I4 = nonmaxsup(I3, or, 1.5);
edgeimage = hysthresh(I4, hithres, lowthres);

% perform the circular Hough transform
h = houghcircle(edgeimage, lradsc, uradsc);

maxtotal = 0;

% find the maximum in the Hough space, and hence
% the parameters of the circle
for i=1:rd

   layer = h(:,:,i);
   [maxlayer] = max(max(layer));


   if maxlayer > maxtotal

      maxtotal = maxlayer;


      r = int32((lradsc+i) / scaling);

      [row,col] = ( find(layer == maxlayer) );


      row = int32(row(1) / scaling); % returns only first max value
      col = int32(col(1) / scaling);

   end

end
```

◆◆◆

```
or = atan2(-Y, X);            % Angles -pi to + pi.
neg = or<0;                   % Map angles to 0-pi.
or = or.*~neg + (or+pi).*neg;
or = or*180/pi;               % Convert to degrees.
```

◆◆◆

```
%
% Usage:  h = addcircle(h, c, radius, weight)
%
% Arguments:
%           h      - 2D accumulator array.
%           c      - [x,y] coords of centre of circle.
%           radius - radius of the circle
%           weight - optional weight of values to be added to the
%                    accumulator array (defaults to 1)
%
% Returns:   h - Updated accumulator array.

function h = addcircle(h, c, radius, weight)

    [hr, hc] = size(h);

    if nargin == 3
    weight = 1;
    end

    % c and radius must be integers
    if any(c-fix(c))
    error('Circle center must be in integer coordinates');
    end

    if radius-fix(radius)
    error('Radius must be an integer');
    end

    x = 0:fix(radius/sqrt(2));
    costheta = sqrt(1 - (x.^2 / radius^2));
    y = round(radius*costheta);

    % Now fill in the 8-way symmetric points on a circle given coords
    % [px py] of a point on the circle.

    px = c(2) + [x  y  y  x -x -y -y -x];
    py = c(1) + [y  x -x -y -y -x  x  y];

    % Cull points that are outside limits
    validx = px>=1 & px<=hr;
    validy = py>=1 & py<=hc;
    valid = find(validx & validy);
```

```
    px = px(valid);
    py = py(valid);

    ind = px+(py-1)*hr;
    h(ind) = h(ind) + weight;
```

◆◆◆

```
x_pupil, y_pupil, r_pupil,eyeimage_filename, radpixels, angulardiv)

global DIAGPATH

radiuspixels = radpixels + 2;
angledivisions = angulardiv-1;

r = 0:(radiuspixels-1);

theta = 0:2*pi/angledivisions:2*pi;

x_iris = double(x_iris);
y_iris = double(y_iris);
r_iris = double(r_iris);

x_pupil = double(x_pupil);
y_pupil = double(y_pupil);
r_pupil = double(r_pupil);

% calculate displacement of pupil center from the iris center
ox = x_pupil - x_iris;
oy = y_pupil - y_iris;

if ox <= 0
   sgn = -1;
elseif ox > 0
   sgn = 1;
end

if ox==0 && oy > 0

   sgn = 1;

end

r = double(r);
theta = double(theta);

a = ones(1,angledivisions+1)* (ox^2 + oy^2);

% need to do something for ox = 0
if ox == 0
   phi = pi/2;
else
```

```
    phi = atan(oy/ox);
end

b = sgn.*cos(pi - phi - theta);

% calculate radius around the iris as a function of the angle
r = (sqrt(a).*b) + ( sqrt( a.*(b.^2) - (a - (r_iris^2))));

r = r - r_pupil;

rmat = ones(1,radiuspixels)'*r;

rmat = rmat.* (ones(angledivisions+1,1)*[0:1/(radiuspixels-1):1])';
rmat = rmat + r_pupil;


% exclude values at the boundary of the pupil iris border, and the iris scelra
border
% as these may not correspond to areas in the iris region and will introduce
noise.
%
% ie don't take the outside rings as iris data.
rmat  = rmat(2:(radiuspixels-1), :);

% calculate cartesian location of each data point around the circular iris
% region
xcosmat = ones(radiuspixels-2,1)*cos(theta);
xsinmat = ones(radiuspixels-2,1)*sin(theta);

xo = rmat.*xcosmat;
yo = rmat.*xsinmat;

xo = x_pupil+xo;
yo = y_pupil-yo;

% extract intensity values into the normalised polar representation through
% interpolation
[x,y] = meshgrid(1:size(image,2),1:size(image,1));
polar_array = interp2(x,y,image,xo,yo);

% create noise array with location of NaNs in polar_array
polar_noise = zeros(size(polar_array));
coords = find(isnan(polar_array));
polar_noise(coords) = 1;
```

```
polar_array = double(polar_array)./255;


% start diagnostics, writing out eye image with rings overlayed

% get rid of outling points in order to write out the circular pattern
coords = find(xo > size(image,2));
xo(coords) = size(image,2);
coords = find(xo < 1);
xo(coords) = 1;

coords = find(yo > size(image,1));
yo(coords) = size(image,1);
coords = find(yo<1);
yo(coords) = 1;

xo = round(xo);
yo = round(yo);

xo = int32(xo);
yo = int32(yo);

ind1 = sub2ind(size(image),double(yo),double(xo));

image = uint8(image);

image(ind1) = 255;
%get pixel coords for circle around iris
[x,y] = circlecoords([x_iris,y_iris],r_iris,size(image));
ind2 = sub2ind(size(image),double(y),double(x));
%get pixel coords for circle around pupil
[xp,yp] = circlecoords([x_pupil,y_pupil],r_pupil,size(image));
ind1 = sub2ind(size(image),double(yp),double(xp));

image(ind2) = 255;
image(ind1) = 255;


% write out rings overlaying original iris image
w = cd;
cd(DIAGPATH);

imwrite(image,[eyeimage_filename,'-normal.jpg'],'jpg');
```

```
cd(w);

% end diagnostics

%replace NaNs before performing feature encoding
coords = find(isnan(polar_array));
polar_array2 = polar_array;
polar_array2(coords) = 0.5;
avg = sum(sum(polar_array2)) / (size(polar_array,1)*size(polar_array,2));
polar_array(coords) = avg;
```

◆◆◆

```
    newtemplate(:,x) = template(:,p+x);

end
```

◆◆

gethammingdistance.m

```
function [HD] = gethammingdistance(template1, template2)

rowcount = size(template1,2);
templatea = zeros(size(template1));
for int = 1:rowcount
    if (template1(1,int) == '1' || template1(1,int)== '0')
    templatea(1,int)= str2num(template1(1,int));
    end
end
rowcount = size(template2,2);
templateb = zeros(size(template2));
for int = 1:rowcount
    if (template2(1,int) == '1' || template2(1,int)== '0')
    templateb(1,int)= str2num(template2(1,int));
    end
end
templatea = logical(templatea);
templateb = logical(templateb);

HD = NaN;

for shifts = -8:8
    template1s = shiftbits(templatea, shifts);

   totalbits = (size(template1s,1)*size(template1s,2));
C = xor(template1s,templateb);

bitsdiff = sum(sum(C==1));

if totalbits == 0
        HD = NaN;
    else

        hd1 = bitsdiff / totalbits;
```

```
        if  hd1 < HD || isnan(HD)
            HD = hd1;
        end
    end
end
```

◆◆◆

APPENDIX D

Data Sheets

List of Data Sheets

1. LM555 Timer
2. LM567 Tone Decoder
3. IR LED (GaAIAs Infrared Emitters: 880nm / SFH485)
4. IrDA (Fast Infrared Transceiver: TFDU5102)
5. LM7808 (Voltage Regulator)

National Semiconductor

July 2006

LM555
Timer

General Description

The LM555 is a highly stable device for generating accurate time delays or oscillation. Additional terminals are provided for triggering or resetting if desired. In the time delay mode of operation, the time is precisely controlled by one external resistor and capacitor. For astable operation as an oscillator, the free running frequency and duty cycle are accurately controlled with two external resistors and one capacitor. The circuit may be triggered and reset on falling waveforms, and the output circuit can source or sink up to 200mA or drive TTL circuits.

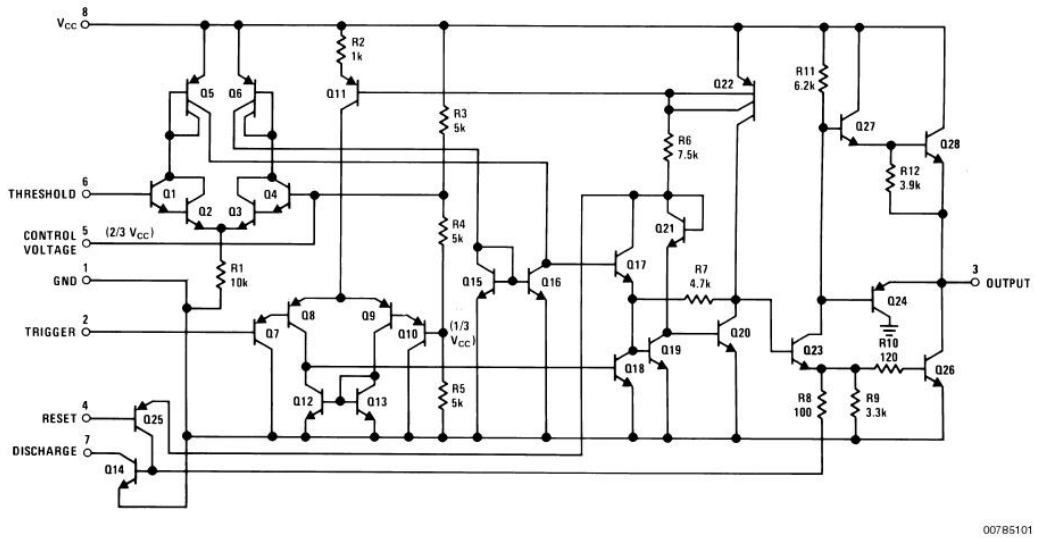
Features

- Direct replacement for SE555/NE555
- Timing from microseconds through hours
- Operates in both astable and monostable modes
- Adjustable duty cycle
- Output can source or sink 200 mA
- Output and supply TTL compatible
- Temperature stability better than 0.005% per ˚C
- Normally on and normally off output
- Available in 8-pin MSOP package

Applications

- Precision timing
- Pulse generation
- Sequential timing
- Time delay generation
- Pulse width modulation
- Pulse position modulation
- Linear ramp generator

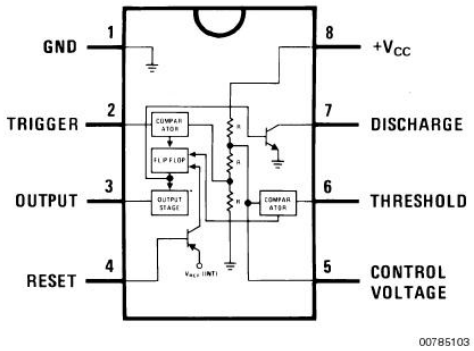
Schematic Diagram



© 2006 National Semiconductor Corporation

Connection Diagram

Dual-In-Line, Small Outline
and Molded Mini Small Outline Packages



Top View

Ordering Information

Package	Part Number	Package Marking	Media Transport	NSC Drawing
8-Pin SOIC	LM555CM	LM555CM	Rails	M08A
	LM555CMX	LM555CM	2.5k Units Tape and Reel	
8-Pin MSOP	LM555CMM	Z55	1k Units Tape and Reel	MUA08A
	LM555CMMX	Z55	3.5k Units Tape and Reel	
8-Pin MDIP	LM555CN	LM555CN	Rails	N08E

Absolute Maximum Ratings (Note 2)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage	+18V
Power Dissipation (Note 3)	
LM555CM, LM555CN	1180 mW
LM555CMM	613 mW
Operating Temperature Ranges	
LM555C	0˚C to +70˚C
Storage Temperature Range	−65˚C to +150˚C

Soldering Information

Dual-In-Line Package	
Soldering (10 Seconds)	260˚C
Small Outline Packages	
(SOIC and MSOP)	
Vapor Phase (60 Seconds)	215˚C
Infrared (15 Seconds)	220˚C

See AN-450 "Surface Mounting Methods and Their Effect on Product Reliability" for other methods of soldering surface mount devices.

Electrical Characteristics (Notes 1, 2)

(T_A = 25˚C, V_{CC} = +5V to +15V, unless othewise specified)

Parameter	Conditions	Limits			Units
		LM555C			
		Min	Typ	Max	
Supply Voltage		4.5		16	V
Supply Current	V_{CC} = 5V, R_L = ∞		3	6	
	V_{CC} = 15V, R_L = ∞		10	15	mA
	(Low State) (Note 4)				
Timing Error, Monostable					
Initial Accuracy			1		%
Drift with Temperature	R_A = 1k to 100kΩ,		50		ppm/˚C
	C = 0.1µF, (Note 5)				
Accuracy over Temperature			1.5		%
Drift with Supply			0.1		%/V
Timing Error, Astable					
Initial Accuracy			2.25		%
Drift with Temperature	R_A, R_B = 1k to 100kΩ,		150		ppm/˚C
	C = 0.1µF, (Note 5)				
Accuracy over Temperature			3.0		%
Drift with Supply			0.30		%/V
Threshold Voltage			0.667		x V_{CC}
Trigger Voltage	V_{CC} = 15V		5		V
	V_{CC} = 5V		1.67		V
Trigger Current			0.5	0.9	µA
Reset Voltage		0.4	0.5	1	V
Reset Current			0.1	0.4	mA
Threshold Current	(Note 6)		0.1	0.25	µA
Control Voltage Level	V_{CC} = 15V	9	10	11	V
	V_{CC} = 5V	2.6	3.33	4	
Pin 7 Leakage Output High			1	100	nA
Pin 7 Sat (Note 7)					
Output Low	V_{CC} = 15V, I_7 = 15mA		180		mV
Output Low	V_{CC} = 4.5V, I_7 = 4.5mA		80	200	mV

Electrical Characteristics (Notes 1, 2) (Continued)

(T_A = 25˚C, V_{CC} = +5V to +15V, unless othewise specified)

Parameter	Conditions	Limits			Units
		LM555C			
		Min	Typ	Max	
Output Voltage Drop (Low)	V_{CC} = 15V				
	I_{SINK} = 10mA		0.1	0.25	V
	I_{SINK} = 50mA		0.4	0.75	V
	I_{SINK} = 100mA		2	2.5	V
	I_{SINK} = 200mA		2.5		V
	V_{CC} = 5V				
	I_{SINK} = 8mA				V
	I_{SINK} = 5mA		0.25	0.35	V
Output Voltage Drop (High)	I_{SOURCE} = 200mA, V_{CC} = 15V		12.5		V
	I_{SOURCE} = 100mA, V_{CC} = 15V	12.75	13.3		V
	V_{CC} = 5V	2.75	3.3		V
Rise Time of Output			100		ns
Fall Time of Output			100		ns

Note 1: All voltages are measured with respect to the ground pin, unless otherwise specified.

Note 2: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. Operating Ratings indicate conditions for which the device is functional, but do not guarantee specific performance limits. Electrical Characteristics state DC and AC electrical specifications under particular test conditions which guarantee specific performance limits. This assumes that the device is within the Operating Ratings. Specifications are not guaranteed for parameters where no limit is given, however, the typical value is a good indication of device performance.

Note 3: For operating at elevated temperatures the device must be derated above 25°C based on a +150°C maximum junction temperature and a thermal resistance of 106°C/W (DIP), 170°C/W (S0-8), and 204°C/W (MSOP) junction to ambient.

Note 4: Supply current when output high typically 1 mA less at V_{CC} = 5V.

Note 5: Tested at V_{CC} = 5V and V_{CC} = 15V.

Note 6: This will determine the maximum value of $R_A + R_B$ for 15V operation. The maximum total ($R_A + R_B$) is 20MΩ.

Note 7: No protection against excessive pin 7 current is necessary providing the package dissipation rating will not be exceeded.

Note 8: Refer to RETS555X drawing of military LM555H and LM555J versions for specifications.

National Semiconductor

February 2003

LM567/LM567C Tone Decoder

General Description

The LM567 and LM567C are general purpose tone decoders designed to provide a saturated transistor switch to ground when an input signal is present within the passband. The circuit consists of an I and Q detector driven by a voltage controlled oscillator which determines the center frequency of the decoder. External components are used to independently set center frequency, bandwidth and output delay.

Features

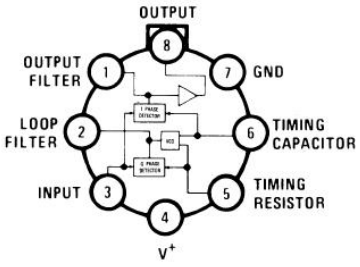
- 20 to 1 frequency range with an external resistor
- Logic compatible output with 100 mA current sinking capability
- Bandwidth adjustable from 0 to 14%
- High rejection of out of band signals and noise
- Immunity to false signals
- Highly stable center frequency
- Center frequency adjustable from 0.01 Hz to 500 kHz

Applications

- Touch tone decoding
- Precision oscillator
- Frequency monitoring and control
- Wide band FSK demodulation
- Ultrasonic controls
- Carrier current remote controls
- Communications paging decoders

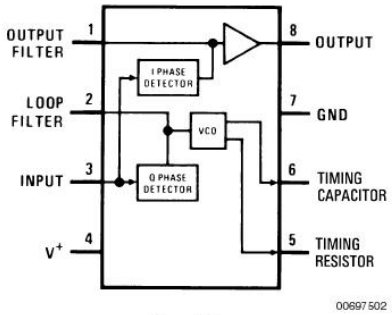
Connection Diagrams

Metal Can Package



Top View
Order Number LM567H or LM567CH
See NS Package Number H08C

Dual-In-Line and Small Outline Packages



Top View
Order Number LM567CM
See NS Package Number M08A
Order Number LM567CN
See NS Package Number N08E

© 2004 National Semiconductor Corporation

Absolute Maximum Ratings (Note 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage Pin	9V
Power Dissipation (Note 2)	1100 mW
V_8	15V
V_3	−10V
V_3	V_4 + 0.5V
Storage Temperature Range	−65˚C to +150˚C
Operating Temperature Range	
LM567H	−55˚C to +125˚C
LM567CH, LM567CM, LM567CN	0˚C to +70˚C
Soldering Information	
Dual-In-Line Package	
Soldering (10 sec.)	260˚C
Small Outline Package	
Vapor Phase (60 sec.)	215˚C
Infrared (15 sec.)	220˚C

See AN-450 "Surface Mounting Methods and Their Effect on Product Reliability" for other methods of soldering surface mount devices.

Electrical Characteristics

AC Test Circuit, T_A = 25˚C, V^+ = 5V

Parameters	Conditions	LM567			LM567C/LM567CM			Units
		Min	Typ	Max	Min	Typ	Max	
Power Supply Voltage Range		4.75	5.0	9.0	4.75	5.0	9.0	V
Power Supply Current Quiescent	R_L = 20k		6	8		7	10	mA
Power Supply Current Activated	R_L = 20k		11	13		12	15	mA
Input Resistance		18	20		15	20		kΩ
Smallest Detectable Input Voltage	I_L = 100 mA, $f_i = f_o$		20	25		20	25	mVrms
Largest No Output Input Voltage	I_C = 100 mA, $f_i = f_o$	10	15		10	15		mVrms
Largest Simultaneous Outband Signal to Inband Signal Ratio			6			6		dB
Minimum Input Signal to Wideband Noise Ratio	B_n = 140 kHz		−6			−6		dB
Largest Detection Bandwidth		12	14	16	10	14	18	% of f_o
Largest Detection Bandwidth Skew			1	2		2	3	% of f_o
Largest Detection Bandwidth Variation with Temperature			±0.1			±0.1		%/˚C
Largest Detection Bandwidth Variation with Supply Voltage	4.75–6.75V		±1	±2		±1	±5	%V
Highest Center Frequency		100	500		100	500		kHz
Center Frequency Stability (4.75–5.75V)	0 < T_A < 70 −55 < T_A < +125		35 ± 60 35 ± 140			35 ± 60 35 ± 140		ppm/˚C ppm/˚C
Center Frequency Shift with Supply Voltage	4.75V–6.75V 4.75V–9V		0.5	1.0 2.0		0.4	2.0 2.0	%/V %/V
Fastest ON-OFF Cycling Rate			f_o/20			f_o/20		
Output Leakage Current	V_8 = 15V		0.01	25		0.01	25	µA
Output Saturation Voltage	e_i = 25 mV, I_8 = 30 mA e_i = 25 mV, I_8 = 100 mA		0.2 0.6	0.4 1.0		0.2 0.6	0.4 1.0	V
Output Fall Time			30			30		ns
Output Rise Time			150			150		ns

Note 1: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. Operating Ratings indicate conditions for which the device is functional, but do not guarantee specific performance limits. Electrical Characteristics state DC and AC electrical specifications under particular test conditions which guarantee specific performance limits. This assumes that the device is within the Operating Ratings. Specifications are not guaranteed for parameters where no limit is given, however, the typical value is a good indication of device performance.

Note 2: The maximum junction temperature of the LM567 and LM567C is 150˚C. For operating at elevated temperatures, devices in the TO-5 package must be derated based on a thermal resistance of 150˚C/W, junction to ambient or 45˚C/W, junction to case. For the DIP the device must be derated based on a thermal resistance of 110˚C/W, junction to ambient. For the Small Outline package, the device must be derated based on a thermal resistance of 160˚C/W, junction to ambient.

Note 3: Refer to RETS567X drawing for specifications of military LM567H version.

Grenzwerte (T_A = 25 °C)
Maximum Ratings

Bezeichnung **Parameter**	**Symbol** **Symbol**	**Wert** **Value**	**Einheit** **Unit**
Betriebs- und Lagertemperatur Operating and storage temperature range	T_{op}; T_{stg}	– 40 … + 100	°C
Sperrspannung Reverse voltage	V_R	5	V
Durchlaßstrom Forward current	I_F	100	mA
Stoßstrom, t_p = 10 µs, D = 0 Surge current	I_{FSM}	2.5	A
Verlustleistung Power dissipation	P_{tot}	200	mW
Wärmewiderstand, freie Beinchenlänge max. 10 mm Thermal resistance, lead length between package bottom and PC-board max. 10 mm	R_{thJA}	375	K/W

Kennwerte (T_A = 25 °C)
Characteristics

Bezeichnung **Parameter**	**Symbol** **Symbol**	**Wert** **Value**	**Einheit** **Unit**
Wellenlänge der Strahlung Wavelength at peak emission I_F = 100 mA	λ_{peak}	880	nm
Spektrale Bandbreite bei 50% von I_{rel} Spectral bandwidth at 50% of I_{rel} I_F = 100 mA	$\Delta\lambda$	80	nm
Abstrahlwinkel Half angle SFH 484 SFH 485	 □ □	 ± 8 ± 20	 Grad deg.
Aktive Chipfläche Active chip area	A	0.09	mm^2
Abmessungen der aktiven Chipfläche Dimension of the active chip area	$L \times B$ $L \times W$	0.3 × 0.3	mm
Abstand Chipoberfläche bis Linsenscheitel Distance chip front to lens top SFH 484 SFH 485	 H H	 5.1 … 5.7 4.2 … 4.8	 mm mm
Schaltzeiten, I_e von 10% auf 90% und von 90% auf 10%, bei I_F = 100 mA, R_L = 50 Ω Switching times, I_e from 10% to 90% and from 90% to 10%, I_F = 100 mA, R_L = 50 Ω	t_r, t_f	0.6/0.5	µs
Kapazität Capacitance V_R = 0 V, f = 1 MHz	C_o	15	pF
Durchlaßspannung Forward voltage I_F = 100 mA, t_p = 20 ms I_F = 1 A, t_p = 100 µs	 V_F V_F	 1.50 (≤ 1.8) 3.00 (≤ 3.8)	 V V
Sperrstrom, Reverse current V_R = 5 V	I_R	0.01 (≤ 1)	µA
Gesamtstrahlungsfluß, Total radiant flux I_F = 100 mA, t_p = 20 ms	Φ_e	25	mW

Kennwerte (T_A = 25 °C)
Characteristics (cont'd)

Bezeichnung **Parameter**	**Symbol** **Symbol**	**Wert** **Value**	**Einheit** **Unit**
Temperaturkoeffizient von I_e bzw. Φ_e, I_F = 100 mA Temperature coefficient of I_e or Φ_e, I_F = 100 mA	TC_I	– 0.5	%/K
Temperaturkoeffizient von V_F, I_F = 100 mA Temperature coefficient of V_F, I_F = 100 mA	TC_V	– 2	mV/K
Temperaturkoeffizient von λ, I_F = 100 mA Temperature coefficient of λ, I_F = 100 mA	TC_λ	0.25	nm/K

Gruppierung der Strahlstärke I_e in Achsrichtung
gemessen bei einem Raumwinkel Ω = 0.001 sr bei SFH 484 bzw. Ω = 0.01 sr bei SFH 485
Grouping of Radiant Intensity I_e in Axial Direction
at a solid angle of Ω = 0.001 sr at SFH 484 or Ω = 0.01 sr at SFH 485

Bezeichnung **Parameter**	**Symbol**	**Wert** **Value**					**Einheit** **Unit**
		SFH 484	**SFH 484-1**	**SFH 484-2**	**SFH 485**	**SFH 485-2**	
Strahlstärke Radiant intensity I_F = 100 mA, t_p = 20 ms	$I_{e\,min}$ $I_{e\,max}$	50 160	50 100	> 80 –	16 80	> 25 –	mW/sr mW/sr
Strahlstärke Radiant intensity I_F = 1 A, t_p = 100 µs	$I_{e\,typ.}$	800	700	900	300	340	mW/sr

Ordering Information

Part Number	Qty / Reel	Description
TFDU5102–TR3	1000 pcs	Oriented in carrier tape for side view surface mounting
TFDU5102–TT3	1000 pcs	Oriented in carrier tape for top view surface mounting

Functional Block Diagram

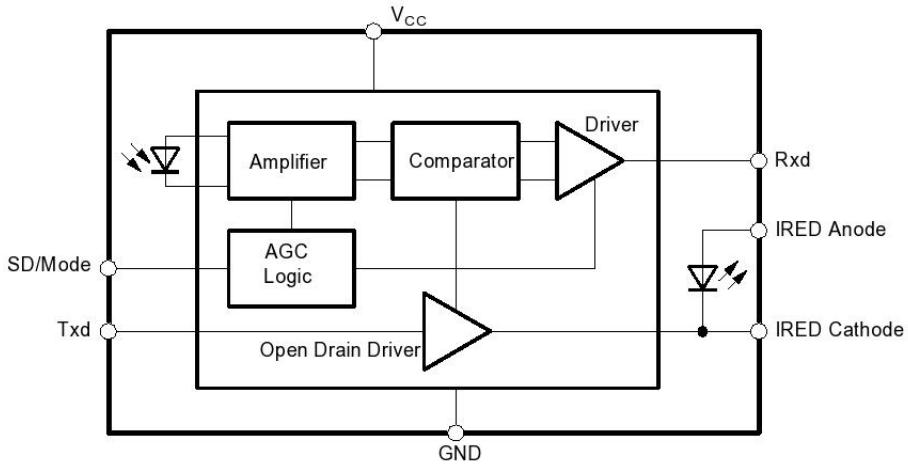


Figure 1. Functional Block Diagram

Pin Description

Pin Number	Function	Description	I/O	Active
1	IRED Anode	IRED anode, to be externally connected to V_{CC} through a current control resistor. This pin is allowed to be supplied from an uncontrolled power supply separated from the controlled V_{CC} supply		
2	IRED Cathode	IRED cathode, internally connected to driver transistor		
3	Txd	Transmit Data Input	I	HIGH
4	Rxd	Pin is floating when device is in shutdown mode	O	LOW
5	SD/Mode	Shutdown/ Mode	I	HIGH
6	V_{CC}	Supply Voltage		
7	Mode	HIGH: High speed mode; LOW: Low speed mode, SIR only (see chapter “Mode Switching”)	I	
8	GND	Ground		

Baby Face (Universal)

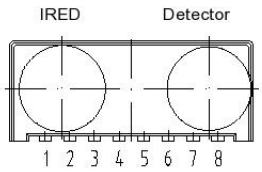


Figure 2. Pinnings

Absolute Maximum Ratings

Reference point Pin GND unless otherwise noted.
Typical values are for DESIGN AID ONLY, not guaranteed nor subject to production testing.

Parameters	Test Conditions	Symbol	Min.	Typ.	Max.	Unit
Supply Voltage Range, Transceiver	0 V <V_{CC2} <6 V	V_{CC1}	– 0.5		6	V
Supply Voltage Range, Transmitter	0 V <V_{CC1} <6 V	V_{CC2}	– 0.5		6	V
Input Currents	For all Pins, Except IRED Anode Pin				10	mA
Output Sinking Current					25	mA
Power Dissipation	See Derating Curve	P_D			350	mW
Junction Temperature		T_J			125	°C
Ambient Temperature Range (Operating)		T_{amb}	–25		+85	°C
Storage Temperature Range		T_{stg}	–25		+85	°C
Soldering Temperature	See Recommended Solder Profile (see Figure 9)				240	°C
Average Output Current		I_{IRED} (DC)			130	mA
Repetitive Pulsed Output Current	<90 µs, t_{on} <20%	I_{IRED} (RP)			600	mA
IRED Anode Voltage		V_{IREDA}	– 0.5		6	V
Transmitter Data Input Voltage		V_{Txd}	– 0.5		V_{CC1} +0.5	V
Receiver Data Output Voltage		V_{Rxd}	– 0.5		V_{CC1} +0.5	V
Virtual Source Size	Method: (1–1/e) encircled energy	d	2.5	2.8		mm
Maximum Intensity for Class 1 Operation of IEC825–1 or EN60825–1 (worst case IrDA FIR pulse pattern)	EN60825, 1997, unidirectional operation, worst case test mode				320	mW/sr

Electrical Characteristics.

T_{amb} = 25°C, V_{CC} = 3.0 V to 5.25 Vunless otherwise noted.
Typical values are for DESIGN AID ONLY, not guaranteed nor subject to production testing.

Parameters	Test Conditions / Pins	Symbol	Min.	Typ.	Max.	Unit
Transceiver						
Supply Voltage		V_{CC}	2.6		5.5	V
Dynamic Supply Current	Receive mode only. In transmit mode, add additional 85 mA (typ) for IRED current					
	SD = Low, E_e = 0 klx	I_{CC}		3	4	mA
	SD = Low, E_e = 1 klx *)	I_{CC}		3	4	mA
Standby Supply Current	SD = High, Mode = Floating, T = 25°C, E_e = 0 klx T = 25°C, E_e = 1 klx *)	I_{SD}			 1 1.5	 µA µA
	SD = High, T = 85°C, Mode = Floating, Not Ambient Light Sensitive	I_{SD}			5	µA
Operating Temperature Range		T_A	–25		+85	°C
Output Voltage Low	R_{load} = 2.2 kΩ, C_{load} = 15 pF	V_{OL}		0.5	0.8	V
Output Voltage High	R_{load} = 2.2 kΩ, C_{load} = 15 pF	V_{OH}	V_{CC}–0.5			V
Input Voltage Low (Txd, SD/ Mode, Mode)		V_{IL}	0		0.8	V
Input Voltage High (Txd, SD/ Mode, Mode)	CMOS level **)	V_{IH}	0.9 x V_{CC}			V
	TTL level, $V_{CC} \geq$ 4.5 V	V_{IH}	2.4			V
Input Leakage Current (Txd, SD/ Mode)		I_L	–10		+10	µA
Input Leakage Current, Mode		I_L	–80		+80	µA
Input Capacitance		C_I			5	pF

*) Standard Illuminant A
**) The typical threshold level is between 0.5 x $V_{CC/2}$ (V_{CC} = 3 V) and 0.4 x V_{CC} (V_{CC} = 5.5 V) .
It is recommended to use the specified min/ max values to avoid increased operating current.

Optoelectronic Characteristics

T_{amb} = 25°C, V_{CC} = 3.0 V to 5.25 V unless otherwise noted.
Typical values are for DESIGN AID ONLY, not guaranteed nor subject to production testing.

Parameters	Test Conditions	Symbol	Min.	Typ.	Max.	Unit
Receiver						
Minimum Detection Threshold Irradiance, SIR Mode	TFDU5102 9.6 kbit/s to 115.2 kbit/s λ = 850 nm to 900 nm	E_e		25	40	mW/m²
Minimum Detection Threshold Irradiance, MIR Mode	TFDU5102 1.152 Mbit/s λ = 850 nm to 900 nm	E_e		65	95	mW/m²
Maximum Detection Threshold Irradiance	λ = 850 nm to 900 nm	E_e	5	10		kW/m²
Logic LOW Receiver Input Irradiance		E_e	4			mW/m²
Rise Time of Output Signal	10% to 90%, @2.2 kΩ, 15 pF	$t_{r\ (Rxd)}$	10		40	ns
Fall Time of Output Signal	90% to 10%, @2.2 kΩ, 15 pF	$t_{f\ (Rxd)}$	10		40	ns
Rxd Pulse Width of Output Signal, 50% SIR Mode	Input pulse length 20 µs, 9.6 kbit/s	t_{PW}	1.2	10	20	µs
	Input pulse length 1.41 µs, 115.2 kbit/s	t_{PW}	1.2		1/2 bit length	µs
Rxd Pulse Width of Output Signal, 50% MIR Mode	Input pulse length 217 ns, 1.152 Mbit/s	t_{PW}	110		260	ns
Stochastic Jitter, Leading Edge, MIR Mode	Input Irradiance = 100 mW/m², 1.152 Mbit/s			±10	±25	ns
Latency		t_L			120	µs
Transmitter						
IRED Operating Current	R1*) = 7.2 Ω, V_{CC} = 5.0 V	I_D		0.4	0.55	A
Output Radiant Intensity (see Figure 3)	V_{CC} = 5.0 V, α = 0°, 15° Txd = High, SD = Low, R1 = 7.2 Ω	I_e	120	170	350	mW/sr
Output Radiant Intensity	V_{CC} = 5.0 V, α = 0°, 15° Txd = Low, SD = High, (Receiver is inactive as long as SD = High) R1 = 7.2 Ω	I_e			0.04	mW/sr
Output Radiant Intensity, Angle of Half Intensity		α		±24		°C
Peak – Emission Wavelength		λ_P	880		900	nm
Optical Output Pulse Duration	Input pulse width 217 ns, 1.152 Mbit/s	t_{opt}	207	217	227	ns
Optical Output Pulse Duration	Input pulse width t < 80 µs Input pulse width t ≥ 80 µs	t_{opt}		t	 80	µs µs
Optical Rise Time, Fall Time		t_{ropt}, t_{fopt}	10		40	ns
Optical Overshoot					10	%

*) R1: control series resistor for current limitation

MC78XX/LM78XX/MC78XXA

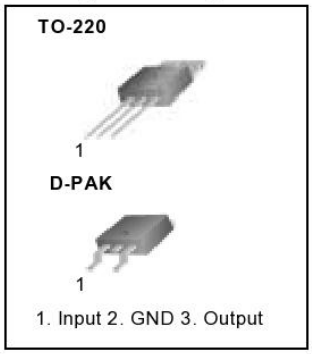
3-Terminal 1A Positive Voltage Regulator

Features

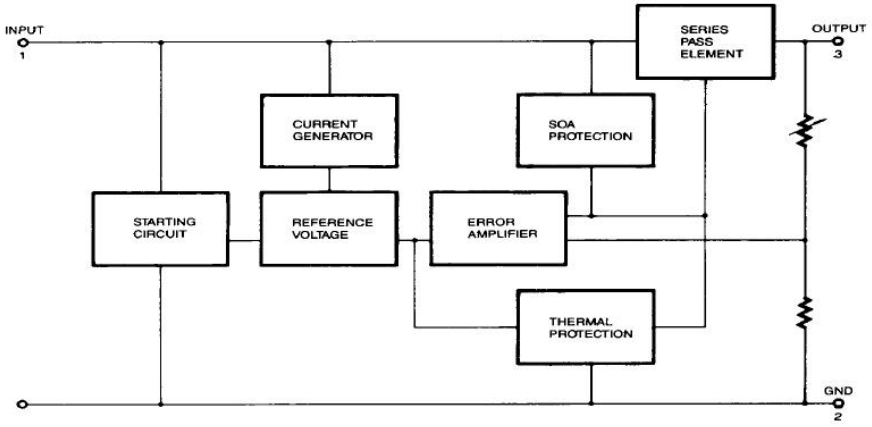
- Output Current up to 1A
- Output Voltages of 5, 6, 8, 9, 10, 12, 15, 18, 24V
- Thermal Overload Protection
- Short Circuit Protection
- Output Transistor Safe Operating Area Protection

Description

The MC78XX/LM78XX/MC78XXA series of three terminal positive regulators are available in the TO-220/D-PAK package and with several fixed output voltages, making them useful in a wide range of applications. Each type employs internal current limiting, thermal shut down and safe operating area protection, making it essentially indestructible. If adequate heat sinking is provided, they can deliver over 1A output current. Although designed primarily as fixed voltage regulators, these devices can be used with external components to obtain adjustable voltages and currents.



Internal Block Digram



Rev. 1.0.1

©2001 Fairchild Semiconductor Corporation

Absolute Maximum Ratings

Parameter	Symbol	Value	Unit
Input Voltage (for V_O = 5V to 18V) (for V_O = 24V)	V_I V_I	35 40	V V
Thermal Resistance Junction-Cases (TO-220)	$R_{\theta JC}$	5	°C/W
Thermal Resistance Junction-Air (TO-220)	$R_{\theta JA}$	65	°C/W
Operating Temperature Range	T_{OPR}	0 ~ +125	°C
Storage Temperature Range	T_{STG}	-65 ~ +150	°C

Electrical Characteristics (MC7805/LM7805)

(Refer to test circuit ,0°C < T_J < 125°C, I_O = 500mA, V_I = 10V, C_I= 0.33μF, C_O= 0.1μF, unless otherwise specified)

Parameter	Symbol	Conditions		MC7805/LM7805			Unit
				Min.	Typ.	Max.	
Output Voltage	V_O	T_J =+25 °C		4.8	5.0	5.2	V
		5.0mA ≤ Io ≤ 1.0A, P_O ≤ 15W V_I = 7V to 20V		4.75	5.0	5.25	
Line Regulation (Note1)	Regline	T_J=+25 °C	V_O = 7V to 25V	-	4.0	100	mV
			V_I = 8V to 12V	-	1.6	50	
Load Regulation (Note1)	Regload	T_J=+25 °C	I_O = 5.0mA to1.5A	-	9	100	mV
			I_O =250mA to 750mA	-	4	50	
Quiescent Current	I_Q	T_J =+25 °C		-	5.0	8.0	mA
Quiescent Current Change	ΔI_Q	I_O = 5mA to 1.0A		-	0.03	0.5	mA
		V_I= 7V to 25V		-	0.3	1.3	
Output Voltage Drift	$\Delta V_O/\Delta T$	I_O= 5mA		-	-0.8	-	mV/ °C
Output Noise Voltage	V_N	f = 10Hz to 100KHz, T_A=+25 °C		-	42	-	μV/Vo
Ripple Rejection	RR	f = 120Hz V_O = 8V to 18V		62	73	-	dB
Dropout Voltage	V_{Drop}	I_O = 1A, T_J =+25 °C		-	2	-	V
Output Resistance	r_O	f = 1KHz		-	15	-	mΩ
Short Circuit Current	I_{SC}	V_I = 35V, T_A =+25 °C		-	230	-	mA
Peak Current	I_{PK}	T_J =+25 °C		-	2.2	-	A

Note:

1. Load and line regulation are specified at constant junction temperature. Changes in V_o due to heating effects must be taken into account separately. Pulse testing with low duty is used.

APPENDIX E

IEEE Article Format

AUTOMATED IRIS RECOGNITION SYSTEM USING CMOS CAMERA WITH PROXIMITY SENSOR

Paulo R. Flores, Hazel Ann T. Poligratis, Angelo S. Victa

School of Electrical, Electronics and Computer Engineering, Mapua Institute of Technology
Muralla St., Intramuros, Manila, Philippines

***Abstract*— Biometrics is becoming popular nowadays due to its very useful security application. These technologies use the unique characteristics of an individual in an electronic system for authentication. There are numbers of biometrics technology and among those; the iris recognition technology is considered the most reliable since human iris is unique and cannot be stolen. The purpose of this design is to improve an existing iris recognition system developed by Engr. Panganiban which is entitled "CCD Camera with Near-Infrared Illumination for Iris Recognition System". The proposed design aims to automate the existing iris recognition system through the use of the following materials: webcam, Gizduino microcontroller, NIR LEDs, power supply, and a proximity sensor. The NIR LEDs, which illuminates the iris, were placed in a circular case attached in the webcam. The iris image that would be captured in this design would only produce little noise since the light produced by the NIR LEDs would be pointing to the pupil of the eye and thus, the iris image template would not be affected. The automation block as its name implies, automates the capturing of the webcam through the use of the sensor, that is connected to the microcontroller in which is handled by the image acquisition software. An additional feature of this design is the real-time processing of image. Once the iris was captured, the software would automatically perform iris segmentation, normalization, template encoding and template matching. It would then display if your iris is authenticated (enrolled) or not. In matching the templates, when the Hamming distance value is greater than or equal to 0.1060, the iris templates do not match but when the HD value is less than 0.1060, the iris template are from the same individual. In comparing the accuracy of the iris templates in our design, the Degrees-of-Freedom (DoF) was computed. The computed DoF of our design is 80, which is higher than that of Engr. Panganiban's work.**

***Keywords*— biometrics, iris recognition, hamming distance, wavelet, real-time-image processing**

I. DESIGN BACKGROUND AND INTRODUCTION

Biometrics is becoming popular nowadays due to its very useful security applications. The technology uses the unique characteristics of an individual in an electronic system for authentication. Biometric technologies, used as a form of identity access management and access control, are becoming the foundation of an extensive array of highly secure identification and personal verification solutions. There are several of applications for biometrics which include civil identity, infrastructure protection, government/public safety and the like. As for the main intention of this design is to implement it for security function since it is very useful to this field having a fact that an iris of a human is the most unique, even for a person, the left iris has different pattern of wavelets compared to that of the right iris of the same person. This design includes an automated CMOS camera and proximity sensor for iris recognition system. A CMOS camera, or complementary metal oxide semiconductor camera, has a CMOS image sensor in which has an ability to integrate a number of processing and control functions. These features include timing logic, exposure control, white balance and the likes. The proximity sensor automates the camera. The sensor decides on whether the target is positioned for capture. The required input information is the iris image of a person for the iris recognition system database. The image will be processed and analyzed by the built-in algorithm in MATLAB. The iris image will be stored in the database as stream of bits. These bits will serve as the identification of the person who enrolled it and will also be used for template matching, a process of finding the owner of the iris template by comparing every iris template in the database.

A. *Statement of the Problem*

The existing Image Acquisition of the Iris Recognition System developed by Panganiban (2009), entitled "CCD Camera with Near-Infrared Illumination for Iris Recognition System" recommends the enhancement of the device to improve the performance of the system. The purpose of this innovation is to answer the following questions:

3. Since quality image affects the critical success of iris image enrolment. What camera should be used to get a better quality image to get a clear detail of the captured iris image?
4. What are the additional components and changes needed, and how can an installation of proximity sensor automate and enhance the precision of the camera and improve the matching rate of accuracy?

B. *Objectives of the Design*

The primary objective of this design is to automate and improve the existing Image Acquisition of the Iris Recognition System by Engr. Panganiban. Specifically, for the success of this design, the following objectives must be met;

6. The Camera to be used, with the help of the NIR LEDs, must be able to produce an image of the subject's iris.
7. NIR LEDs must be located where it would give enough IR light to the subject's iris. This would help make the iris more visible to the camera and to the image for capture.
8. The Proximity sensor should be installed to the system which would detect whether the person is at the correct distance and position before capturing the subject's iris.
9. The system must be able to recognize the difference between the irises to be processed through Hamming distance values and show the separation of classes through degree-of-freedom (DoF).
10. The system must have a DoF improvement on Engr. Panganiban's design.

C. *Impact of the Design*

The design is an Automated Iris Recognition System; it is generally made for improving its image acquisition. This would capture an image of the iris. Nowadays, this biometric technology shows an increasing promise on the security system for it studies the unchanging measurable biological characteristics that are unique to each individual. Among the existing biometric devices and scanners available today, it is generally conceded that iris recognition is the most accurate. The design can be used as a prototype which can be implemented by companies, governments, military, banks, airports, research laboratories, border control for security purposes for allowing and limiting access to a particular information or area. The government officials could also use this design for identifying and recording information of individuals and criminals.

Iris recognition technology can be used in places demanding high security. Physical access-based identification, which includes anything requiring a password, personal identification number or key for building access or the like, could be replaced by this technology. Unlike those physical methods of identification, human iris cannot be stolen. This technology addresses the problems of both password management and fraud.

D. *Design Constraints*

Good quality iris image can only be produced if the eye is approximately 3 to 4 cm away from the camera. A solid red light from the proximity sensor would indicate that the human eye is within the range of 4 to 5 cm. Every time an object is sensed, the red LED generates a solid light and the camera captures an image of the object. The system does not involve iris image processing and matching of individuals with eye disorders or contact lenses. Since with these situations, the iris image will be affected. Also, the system will only work properly when the captured image is an iris otherwise it will result to an error. The speed of the system is limited by the computer specifications where the software is deployed. The recommended system requirements for the software application is a multi-core 2.20 GHz or higher for the CPU, a 4.00 GB or higher for the RAM and Windows 7 for the operating system.

II. REVIEW OF RELATED LITERATURE AND STUDIES

Iris Recognition Technology

Biometrics became popular in security applications due to its personal identification and verification based on the physiological and behavioural characteristics of the subject. Among the existing biometric technologies, it is iris recognition that is considered promising which uses the apparent pattern of the human iris (Panganiban, 2010). The iris is a muscle within the eye that regulates the size of the pupil which controls the amount of light that enters the eye. It is the colored portion of the eye with coloring based on the amount of melatonin pigment within the muscle. The coloration and structure of the iris is genetically linked but the details of the patterns are not (National Science and Technology Council, 2006).

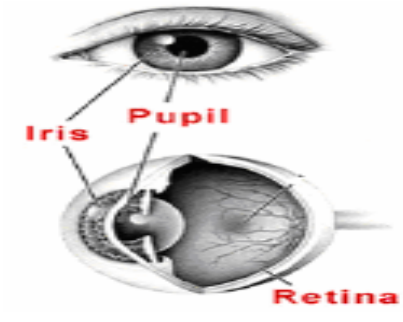


Figure 2.1 Iris Diagram

Irises contain approximately 266 distinctive characteristics, about 173 of which are used to create the iris template and serves as a basis for biometric identification of individuals. Iris patterns possess high inter-class dependency, and low intra-class dependency (Daugman, 1993).

Image Quality

According to Kalka, et al., the performance of the iris recognition system, particularly recognition and segmentation, and the interoperability are highly dependent in the quality of the iris image. There are different factors that affect the image quality namely defocus blur, motion blur, off-angle, occlusion, lighting, specular reflection, and pixel-counts.

The camera must possess excellent imaging performance in order to produce accurate results. In a CMOS (Complementary Metal Oxide Semiconductor) Image sensor, each pixel has its own charge-to-voltage conversion. CMOS image sensor often includes amplifiers, noise-correction, and digitalization circuits, so that the chip outputs digital bits. Because of these features, the design complexity increases and the area available for light capture decreases.

Iris Image Quality Metrics

Iris Image Quality Document, in Part 6 of ISO/IEC 29794, establishes terms and definitions that are useful in the specification, characterization and test of iris image quality. Some of the common quality metrics for iris images are the following: Sharpness, Contrast, Gray scale density, Iris boundary shape, Motion blur, Noise and Usable Iris Area.

Sharpness is the factor which determines the amount of detail an image can convey. It is affected by the lens, particularly the design and manufacturing quality, focal length, aperture, and distance from the image center, as well as the sensor (pixel count and anti-aliasing filter). In the field, sharpness is affected by camera shake, focus accuracy, and atmospheric disturbances like thermal effects and aerosols. Lost sharpness can be restored by sharpening, but sharpening has limits. Over sharpening can degrade image quality by causing halos to appear near contrast boundaries.

Dynamic range (or exposure range) is the range of light levels a camera can capture, usually measured in f-stops, Exposure Value, or zones. It is closely related to noise: high noise implies low dynamic range.

Contrast, also known as gamma, is the slope of the tone reproduction curve in a log-log space. High contrast usually involves loss of dynamic range — loss of detail, or clipping, in highlights or shadows.

Motion blur is the apparent streaking of rapidly moving objects in a still image or a sequence of images. This results when the image being captured changes during the grabbing of a single frame, either due to rapid movement or long exposure.

Pixel resolution is often used for a pixel count in digital imaging. An image of N pixels high by M pixels wide can have any resolution less than N lines per picture height, or N TV lines. But when the pixel counts are referred to as resolution, the convention is to describe the pixel resolution with the set of two positive integer numbers, where the first number is the number of pixel columns (width) and the second is the number of pixel rows (height), for example as 640 by 480. Another popular convention is to cite resolution as the total number of pixels in the image, typically given as number of megapixels, which can be calculated by multiplying pixel columns by pixel rows and dividing by one million.

According to the same standards, the number of effective pixels that an image sensor or digital camera has is the count of elementary pixel sensors that contribute to the final image, as opposed to the number of total pixels, which includes unused or light-shielded pixels around the edges.

Image noise is the random variation of brightness or color information in images produced by the sensor and circuitry of a scanner or digital camera. Image noise can also originate in film grain and in the unavoidable shot noise of an ideal photon detector. It is generally regarded as an undesirable by-product of image capture. According to Makoto Shohara, noise is dependent on the background color and luminance. They conducted subjective and quantitative experiments for three noise models, using a modified grayscale method. The subjective experiment results showed the perceived color noise depends on the background color, but the perceived luminance noise does not.

Proximity Sensor

A proximity sensor detects the presence of nearby objects without any physical contact. This type of sensor emits a beam of electromagnetic radiation, such as infrared, and looks for changes in the field or a return signal. The proximity sensor automates the camera by deciding on whether the target is positioned for capture.

Iris Image Acquisition

Image acquisition depends highly on the image quality. According to Dong, et al. (2008), the average iris diameter is averagely 10 millimeters, and the required pixel number in iris diameter is normally more than 150 pixels in iris image acquisition systems. The International standard regulates that 200 pixels is of “good quality”, 150-200 is “acceptable quality” and 100-150 is “marginal quality”. The iris image with a smaller pixel is considered as of a better quality image and a bigger pixel as of less quality image.

In Panganiban’s study (2010), it was mentioned that Phinney and Jelinek have claimed that near-infrared illumination is safe to the human eye. Derwent Infrared Illuminators supported the safeness of near-infrared illumination to the eye. Studies showed that filtered infrared is approximately 100 times less hazardous than the visible light.

Iris Recognition System and Principles

Libor Masek's proposed algorithm showed an automatic segmentation algorithm which localise the iris region from an eye image and isolate eyelid, eyelash and reflection areas. The circular Hough transform, which localised the iris and pupil regions, was used for the automatic segmentation and the linear Hough transform was used for localising occluding eyelids. Thresholding was performed for the isolation of the eyelashes and reflections. The segmented iris region was normalised by implementing Daugman's rubber sheet model. The iris is modelled as a flexible rubber sheet, which was unwrapped into a rectangular block with constant polar dimensions to eliminate dimensional inconsistencies between iris regions. Then the features of the iris were encoded by convolving the normalised iris region with 1D Log-Gabor filters and phase quantising the output in order to produce a bit-wise biometric template. The Hamming distance was chosen as a matching metric. This gave a measure on the number of bits that disagreed between two templates. A failure of statistical independence between two templates would result in a match. This means that the two templates were considered to have been generated from the same iris if the Hamming distance produced was lower than a set Hamming distance.

In the proposed algorithm of Panganiban (2010), the feature vector was encoded using Haar and Biorthogonal wavelet families at various levels of decomposition. Vertical coefficients were used for implementation because of the dominant features of the normalized images that were oriented vertically. Hamming distance was used to define the inter-class and intra-class relationships of the templates. The computed number of degrees of freedom which was based on the mean and the standard deviation of the binomial distribution demonstrated the separation of iris classes. Proper choice of threshold value is needed in the success of the iris recognition. But if there were instances where a clear decision cannot be made based on a preset threshold value, the comparison between the relative values of Hamming distances can lead to correct recognition. The determination of identity in her study was based on both the threshold value and on a comparison of HD values. The test metrics proved that her proposed algorithm has a high recognition rate.

Biometric Test Metrics

Ives, et al. (2005) determined the consequences of compression through the analysing the compression rate. Also, each pair of curves (False Rejection Rate (FRR) and False Accept Rate (FAR)) represents the comparison of each compressed database against the original database. An original versus original comparison is included as a baseline. The compression ratio increases, the FAR curve remains virtually unchanged, while the FRR curves move further to the right which causes an increased Equal Error Rate (EER, where FAR = FRR), and an increased number of errors (False Accepts + False Rejects) which reduces overall system accuracy.

Sarhan (2009) compares the iris images by using the Hamming distance which provides a measure as to how many bits are the same between two patterns. The number of degrees of freedom represented by the templates measures the complexity of iris patterns. This was measured by approximating the collection of inter-class Hamming distance values as binomial distribution. FAR (False Accept Rate) is the probability that the system incorrectly matches the input pattern to the non-matching template in the database. The FRR (False Reject Rate) is the probability that the system fails to detect a match between the input pattern and a matching template in the database. The ROC (Relative Operating Characteristic) plot is the visual characterization of the trade-off between the FAR and FRR. The EER (Equal Error Rate) is the rate at which both accept and reject errors are equal.

Panganiban (2010) determined the performance of each feature of the vector in terms of the accuracy over vector length. The threshold values were identified through the range of the Hamming distance. Poor Quality means that the Hamming distance value is 10 % lower than the threshold value. Moderate Quality means that the user has to decide whether the Hamming distance value agrees with the desired result. This occurs when the value is ± 10 % of the threshold values. Good Quality means that the Hamming 40 distance value is 10% higher than the threshold value. False Accept Rate (FAR) is the probability that the system accepts an unauthorized user or a false template which is computed using the formula $FAR = P_{inter}/n$, where P_{inter} is the number of HD values that fall under Poor Quality of the inter-class distribution and n is the total number of samples. False Reject Rate (FRR) is the probability that the system rejects an authorized user or a correct template which is computed using the formula $FRR = P_{intra}/n$, where P_{intra} is the number of HD values that fall under Poor Quality of the intra-class distribution and n is the total number of samples. The Equal Error Rate (EER) compares the accuracy of devices. The lower the EER, the more accurate the system is considered to be. The characteristic of the wavelet transform are the concept used in encoding iris bit patterns. These metrics are useful in achieving the accuracy and efficiency of wavelet coefficients.

III. DESIGN PROCEDURES

A. Hardware Development

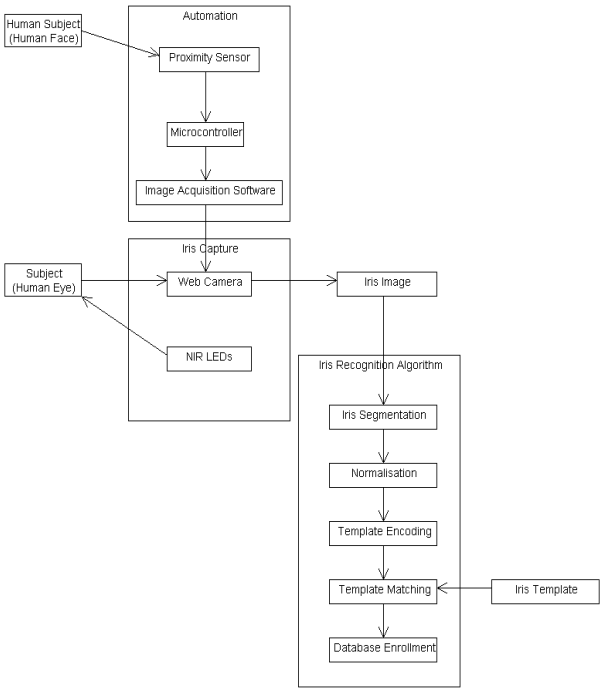


Figure 3.1 Block Diagram

The block diagram of the design is shown in figure 3.1. The automation part is composed of the proximity sensor, the microcontroller and the image acquisition software. This automation block as its name implies, automates the capturing of the webcam through the use of the sensor, that is connected to the microcontroller in which is handled by the image acquisition software. The proximity sensor senses objects within 10cmrange from its transceiver. The microcontroller used is the Gizduino microcontroller manufactured and produced by E-Gizmo. The image acquisition software is developed using MATLAB R2009a. The next part is the Iris Capture block. It consists of the webcam and the NIR LEDs. The webcam is connected to the computer through its USB cord. The NIR LEDs are the one responsible for the visibility of the iris to the webcam. If the image acquisition software tells the webcam to capture, the webcam will do so and an iris image will be produced. The final part is the iris recognition algorithm. The iris recognition algorithm starts with the iris segmentation process. It is based on the circular Hough transform which is similar to the equation of a circle ($X_C^2 + Y_C^2 = r^2$). Since the iris of the eye is ideally shaped like a circle, the Hough transform is used to determine the properties of geometric objects found in an image like circles, and lines. Canny edge detection is used to detect edges of shapes. It is developed by John F. Canny in 1986. Horizontal lines are drawn on the top and bottom eyelid to separate the iris and two circles are drawn, one for the pupil and the other one for the iris. The value of the iris radius to be used ranges from 75 to 85 pixels and for the pupil radius ranges from 20 to 60 pixels. After the iris is segmented, it is normalized. In normalization, the segmented iris is converted to a rectangular shaped-strip with fixed dimensions. This process uses Daugman's rubber sheet model. The image will then be analyzed using 2D wavelets at maximum level of 5. After that, a biometric template is produced. Similar to Engr. Panganiban's work, the wavelet transform is used to extract the discriminating information in an iris pattern. Only one mother wavelet is used which is the Haar because it produced the highest CRR according to Engr. Panganiban's thesis. The template is encoded using the patterns that yielded during the wavelet decomposition. Then, the algorithm will check if the template matches another template stored in the database by using its binary form to compute for the hamming distance of the two templates. This is done by using the XOR operation. A template can also be added to the database by using MS SQL queries.

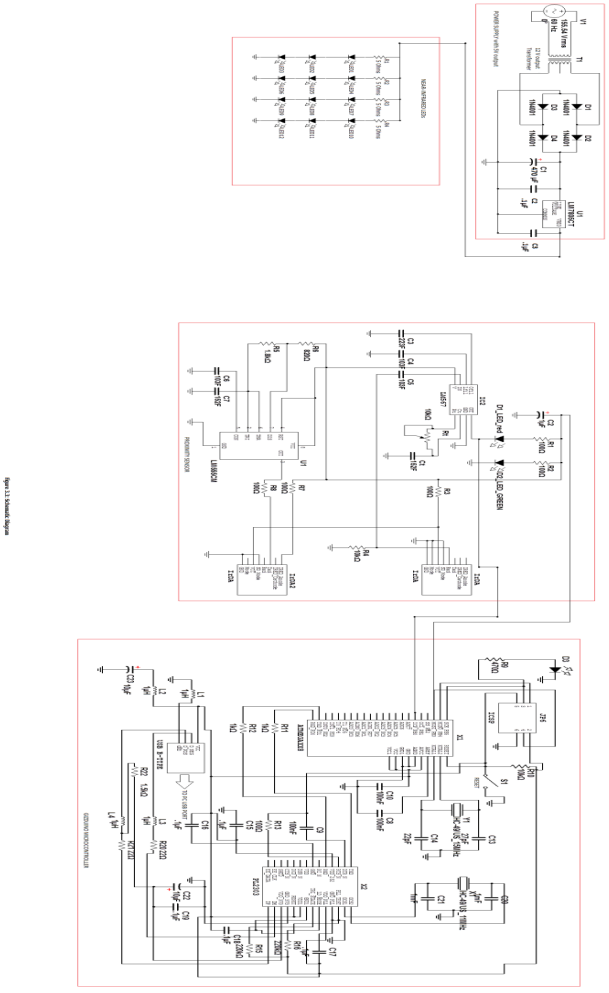


Figure 3.2 Schematic Diagram

Figure 3.2 shows the design's schematic diagram. The Near Infrared LEDs serves as the lighting source. The light produced by the near-infrared diodes is only visible in the camera and not with the human eye. It produces less noise in the image when captured than visible light. The resistors used each have 5-ohms resistance. This was computed using the formula:$R = (V_S - V_F) / I_F$ where V_S is the voltage source of 5-V, V_F is the voltage drop of 1.5-V and an I_F is a current of 100-mA. The formula would produce a resistance of 35-ohms. But considering that we are to connect in parallel four rows of 3 NIR LEDs in series, the resulting resistance value 'R' connected in series with the 3 NIR LEDs on each row would be 5-ohms.

The proximity sensor detects the presence of nearby objects without any physical contact. This type of sensor emits a beam of electromagnetic radiation, such as infrared, and looks for changes in the field or a return signal. This gives the appropriate signal to the image-capturing software when the subject is in the right position for iris image acquisition.

The Gizduino microcontroller is a clone of Arduino microcontroller made by the company E-Gizmo. It has a built-in ATMEGA microcontroller and PL2303 USB to RS-232 Bridge Controller.

B. Software Development

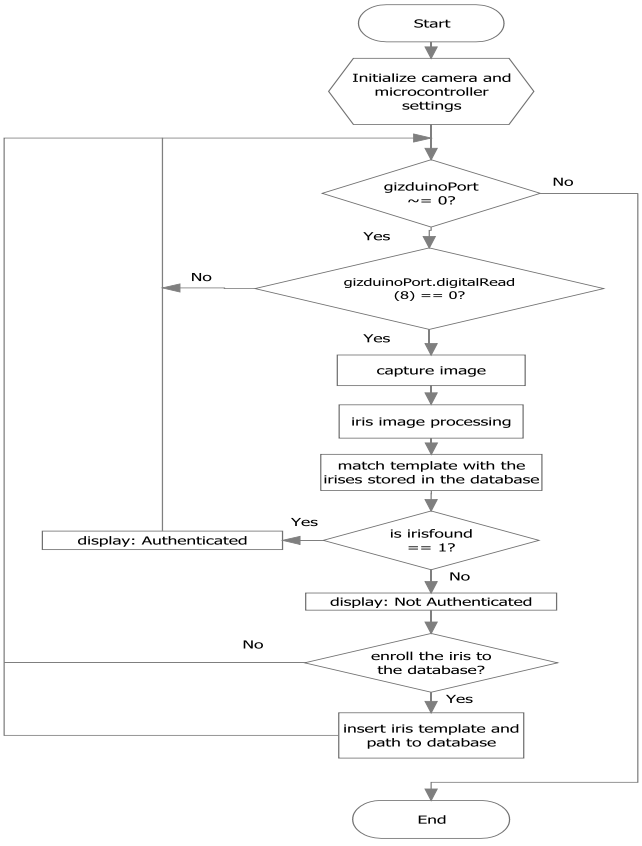


Figure 3.3 System Flowchart

Figure 3.3 illustrates the flowchart of the system. First, the system initializes the camera and the microcontroller settings. Then, it checks whether the Gizduino microcontroller is connected or not by checking the value of gizduinoPort. While it is equal to zero, the system will end its process. But while its value is not equal to zero, meaning the MCU is still connected, it inspects if the person's face is within the correct distance by checking the value of gizduinoPort.digitalRead(8). If the value is zero, it means that the distance is correct according to the proximity sensor and the program triggers the camera to capture the iris image. After capturing the image, the system processes it, extracts the iris feature and encodes the template into bits. After that, the system compares the encoded template with all the templates stored in the database. When a match is found, the program displays a message box telling that the person's iris is authenticated and is registered on the database and then the system prepares for the next capture by going back to the distance inspection. But when it's not found, the program displays a message box again however telling that it is not found and it's not authenticated. Also, the system asks if the unauthenticated iris template is to be enrolled in the database or not. If it would be enrolled, then the iris template and its path are inserted into the database and then the system goes back to the distance inspection. Else if it's not to be enrolled, then the system just goes back to the distance inspection.

IrisDataBankDesign			
Column Name	**Data Type**	**Key Type**	**Allow Null**
IrisId	Int	PK	No
IrisPath	varchar(50)	NONE	Yes
IrisTemplate	varchar(MAX)	NONE	Yes

Figure 3.4 Relational Model

The template bits are stored in a database using Microsoft SQL 2005 Express edition. In Fig. 3.4, the IrisId field is set to auto-increment by 1 and the primary key. While the IrisPath and IrisTemplate depends on the output of the system which is inserted to the database.

C. Prototype Development

Quantity	Material	Description
1 pc	5-V 750-mA Power Supply	Powers up the NIR LEDs
12 pcs	NIR LEDs	Illuminates the iris
1 pc	Proximity Sensor	Senses if the iris is within the detecting range
1 pc	Gizduino Microcontroller	Implements the designed program
1 pc	Webcam	Captures the iris image

IV. TESTING, PRESENTATION, AND INTERPRETATION OF DATA

Automated CMOS Camera for iris recognition through proximity sensor focus on its objective of improving an existing image acquisition of the iris recognition system developed by Engr. Panganiban and the design's automation. In this chapter, the researchers conduct experiments to identify whether the hardware and software design meet the criteria for an effective iris recognition system. Several observations and assessments are provided, together with reliable measurements or data that will support the researcher's remarks.

SENSOR OUTPUT TEST

The proximity sensor automates the system by detecting whether the person is at the correct distance and position before capturing the subject's iris. Further testing on the proximity sensor was done because there has been a suspected glitch found on the proximity sensor.

SETTINGS	
Position:	Placed on top of the camera
Input:	Person's forehead

Table 4.1 Proximity Sensor Settings

DISTANCE(cm)	Red LED Status (Output)
1	Solid Red Light
2	Solid Red Light
3	Solid Red Light
4	Solid Red Light
5	Flickering Red Light
6	No light
7	No light
8	No light
9	No light
10	No light

Table 4.2 Sensor Output Testing

As seen in Table 4.2, the correctness of the distance and position was seen on the red LED's intensity with respect to the settings indicated in table 4.1. A solid red light was seen when an object is 0cm to 4m away from the IrDA. But a flickering red light was seen when the range is within the range of 4cm to 5cm. The LED does not produce light when the object is greater than 5cm. Also, these findings were relevant to the behaviour of the camera. When the red LED has a solid light, the camera captures every time an object is sensed.

IMAGE QUALITY TEST

The performance of the iris recognition system, particularly recognition and segmentation, and the interoperability are highly dependent in the quality of the iris image.

Specifications	CCD Camera	A4tech PK 710mj live messenger 5M Webcam
Image Sensor	CCD image sensor with validity pixel of PAL: 512x528/- 512x492	CMOS image sensor, 640x480pixels
Focus Range	Manual focus according to user requirement	Manual Focus, 2cm to infinity (according to user requirement)

Table 4.3 Camera Specifications

Our group replaced Eng'r. Panganiban's CCD Camera with a CMOS Camera. The camera must possess excellent imaging performance in order to produce accurate results. In a CCD (Charge Couple Device) sensor, every pixel's charge is transferred through a very limited number of output nodes to be converted to voltage, buffered, and sent off-chip as an analog signal. All of the pixel can be devoted to light capture, and the uniformity of the output is high. In a CMOS (Complementary Metal Oxide Semiconductor) sensor, each pixel has its own charge-to-voltage conversion, and the sensor often includes amplifiers, noise-correction, and digitalization circuits, so that the chip outputs digital bits. With these, the design complexity increases and the area available for light capture decreases. The uniformity is lower because each pixel is doing its own conversion. Also, both cameras that were used were manual focus, for the user to adjust it to their system's requirements.

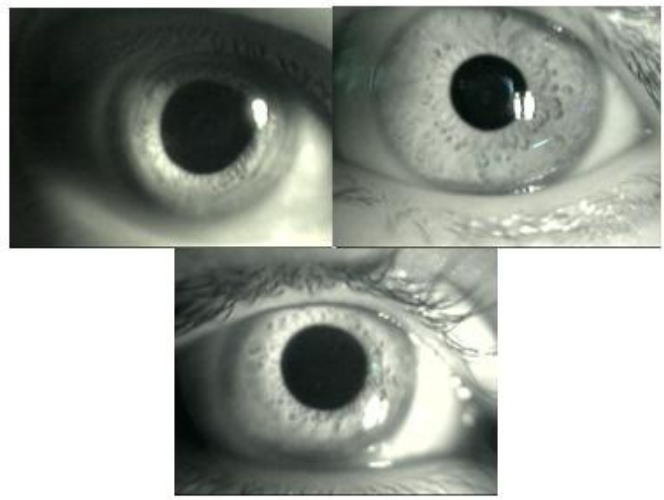


Figure 4.1 Selected Iris Images from Engr. Panganiban's system

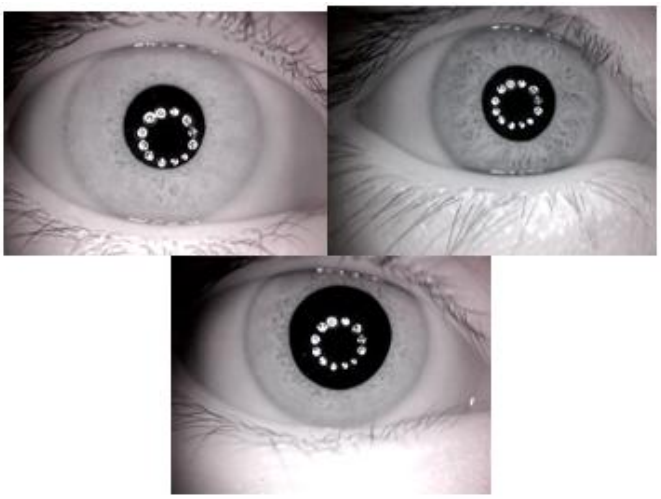


Figure 4.2 Selected Iris Images from the Current System

Common Quality Metrics	Figure 4.1	Figure 4.2
Blur Motion	Blurred Image	Clear Image
Noise in the Iris Image	With Noise	Without Noise
Brightness	Dark	Bright
Magnification	Blurred Image	Clear Image

Table 4.4 Iris Image Quality Assessment

In Table 4.4, it can be observed that the improved design really showed promising results. The design produced a clear and bright image even though the image was magnified in the test. The magnification testing was made by zooming in the images. Also, there was no noise in the iris image.

ID Number	Iris Image
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	

Table 4.5 Enrolled Captured Iris Images

DATASETS

In Table 4.5, the iris images that were captured and enrolled into the Iris Recognition System are displayed. These images undergone image processing as discussed in the previous chapter to have its iris template be produced. The iris templates were encoded using the Haar mother wavelet because according to Engr. Panganiban's work, it resulted with the best values of Hamming distance after every iris template were compared. The Inter-class comparisons of Haar wavelet at Level 4 vertical coefficient is shown on Table 4.6.

As seen on the table, the maximum HD value is 0.1538 and the minimum is 0.1060. A zero value indicates that the iris templates are perfectly matching each other.

Iris Id	1	2	3	4	5	6	7	8	9	10
1	0.0000	0.1331	0.1331	0.1268	0.1060	0.1268	0.1227	0.1331	0.1081	0.1206
2	0.1331	0.0000	0.1331	0.1518	0.1351	0.1268	0.1227	0.1372	0.1372	0.1414
3	0.1331	0.1331	0.0000	0.1268	0.1351	0.1227	0.1268	0.1081	0.1247	0.1372
4	0.1268	0.1518	0.1268	0.0000	0.1247	0.1372	0.1206	0.1143	0.1227	0.1060
5	0.1060	0.1351	0.1351	0.1247	0.0000	0.1123	0.1538	0.1435	0.1351	0.1310
6	0.1268	0.1268	0.1227	0.1372	0.1123	0.0000	0.1289	0.1393	0.0977	0.1268
7	0.1227	0.1227	0.1268	0.1206	0.1538	0.1289	0.0000	0.1227	0.1102	0.1185
8	0.1331	0.1372	0.1081	0.1143	0.1435	0.1393	0.1227	0.0000	0.1123	0.1206
9	0.1081	0.1372	0.1247	0.1227	0.1351	0.0977	0.1102	0.1123	0.0000	0.1206
10	0.1206	0.1414	0.1372	0.1060	0.1310	0.1268	0.1185	0.1206	0.1206	0.0000

Table 4.6 Inter-class Comparisons of Haar Wavelet at Level 4 Vertical Coefficient

It is observable that when the Hamming distance value is greater than or equal to 0.1060, the iris templates do not match. In table 4.6, the Intra-class comparisons of Haar Wavelet at level 4 vertical coefficient shows that when the HD value is less than 0.1060, the iris template are from the same individual. Using the formula for the degrees of freedom:

$$DOF = \frac{p(1-p)}{\sigma^2}$$

Where p is the mean which is equal to 0.1261 and the σ is the standard deviation which is equal to 0.03954, the number of degrees of freedom is 80. According to statistics, this is the number of degrees of freedom that the values in this case, the HD values are free to vary.

Iris Id	1	2	3	4	5	6	7	8	9	10
1	0.0811	0.1351	0.1310	0.1060	0.1123	0.1247	0.1227	0.1289	0.1185	0.1227
2	0.1310	0.0603	0.1019	0.1185	0.1331	0.1123	0.1268	0.1289	0.1435	0.1227
3	0.1310	0.1310	0.0686	0.1351	0.1372	0.1247	0.1476	0.1081	0.1227	0.1019
4	0.1164	0.1331	0.1497	0.0956	0.1268	0.1393	0.1247	0.1310	0.1123	0.104
5	0.1247	0.1206	0.1455	0.1247	0.0852	0.1060	0.1372	0.1351	0.1206	0.1518
6	0.1372	0.1206	0.1081	0.1289	0.1185	0.0520	0.1372	0.1227	0.0915	0.1372
7	0.1164	0.1164	0.1164	0.1080	0.1310	0.1393	0.0873	0.1185	0.1164	0.1206
8	0.1310	0.1518	0.1268	0.1310	0.1580	0.1060	0.1227	0.0748	0.1227	0.1227
9	0.1102	0.1435	0.1185	0.1019	0.1289	0.1123	0.1435	0.1164	0.0561	0.1143
10	0.1019	0.1227	0.1476	0.1143	0.1455	0.1247	0.0977	0.1247	0.1143	0.0977

Table 4.7 Intra-class Comparisons of Haar Wavelet at Level 4 Vertical Coefficient

A. Impact Analysis

The iris recognition system of Engr. Panganiban was taken to the next level by adding real time image processing features to it. This would be easier to use for the user would just look into the camera and wait for just a short period of time for the system to capture and process his or her iris. After the image was processed, it would immediately display if the person is authenticated or not.

The designed iris recognition showed an increasing promise on the security system for it analyses the unchanging measurable biological characteristics that are unique to each individual. The design can be used as a prototype which can be implemented by in places demanding high security such as companies, governments, military, banks, airports, research laboratories and border control area. This would allow and limit access to a particular information or area. The government officials could also use this design for identifying and recording information of individuals and criminals. Physical methods of identification, which includes anything requiring a password, personal identification number or key for building access or the like, are easily hacked or stolen but human iris cannot be stolen. This technology addresses the problems of both password management and fraud.

V. CONCLUSIONS AND RECOMMENDATION

This chapter gives the overall conclusion of the design covering up all the objectives specified in Chapter1. This chapter also tackles the important results of the test performed in Chapter 4 including the limitations of the design. The recommendation part of this chapter suggests what should be done to improve the design.

A. Conclusion

Based from the results obtained, the design was proven sufficient for iris recognition. The camera used is a manual focus- CMOS camera. In a Complementary Metal Oxide Semiconductor sensor, each pixel has its own charge-to-voltage conversion, and the sensor often includes amplifiers, noise-correction, and digitalization circuits, so that the chip outputs digital bits. With these, the design complexity increases and the area available for light capture decreases. The correct positioning of the webcam, NIR LEDs and sensor produced a clearer and brighter iris image which really improves the performance of the iris recognition system. The NIR LEDs must be attached circular to the webcam so that noise that would be produced in the iris would be lessened. The light of the NIR LEDs would be directed to the pupil. Since the light reflection will be located in the pupil, it would not affect the iris segmentation and that the iris template. The case of the camera also lessens the noise since it blocks other factors that might affect the iris image and results.

The proximity sensor has a delay of 5 seconds before it sends signal for the webcam to capture the iris image. There is a delay so that the user can position his or her eye properly to the device.

Also, the results showed that when the Hamming distance value is greater than or equal to 0.1060, the iris templates do not match. The Intra-class comparison of Haar Wavelet at level 4 vertical coefficient shows that when the HD value is less than 0.1060, the iris templates are from the same individual.

From the results of the Hamming Distance in inter-class comparison, the Degrees of Freedom (DoF) computed is 80,

which is higher than of Engr. Panganiban's work which is equal to 50. This shows that the comparison of iris templates in our design is more accurate.

B. Recommendation

Although the obtained results proved that the design is sufficient for iris recognition, the following are still recommended for the improvement of the system's performance:

1. The proximity sensor may be replaced by an algorithm such as pattern recognition that will allow the software to capture the iris image once a circular shape is near the camera.

2. The digital camera can be converted to an Infrared Camera which would replace the webcam and NIR LEDs.

3. Artificial Intelligence, such as Fuzzy Logic, can be applied to the system to improve the performance of the Iris recognition system.

4. Embedding the Iris recognition system, its hardware and software into one device can be done to have the speed of the system independent on the speed of the computer used and could also be portable.

REFERENCES

[1] Addison, P. (2002). The Illustrated Wavelet Transform Handbook, Institute of Physics.

[2] Bradley J., Brislawn, C., and Hopper, T. (1993). The FBI Wavelet/Scalar Quantization Standard for Gray-scale Fingerprint Image Compression. Tech. Report LA-UR-93-1659, Los Alamos Nat'l Lab, Los Alamos, N.M.

[3] Boles, W.W. and Boashash, B.A. (1998). A human identification technique using images of the iris and wavelet transform," IEEE trans. on signal processing, vol. 46, issue 4.

[4] Canny, J. (1986). A Computational Approach To Edge Detection, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679–714.

[5] Cohn, J. (2006). Keeping an Eye on School Security: The Iris Recognition Project in New Jersey Schools. NIJ Journal, no. 254.

[6] Huifang, H. and Guangshu, H. (2005). Iris recognition based on adjustable scale wavelet transform. Proceedings of the 2005 IEEE.

[7] Kong, W. and Zhang, D. (2001). Accurate iris segmentation based on novel reflection and eyelash detection model. Proceedings of 2001 International Symposium on Intelligent Multimedia, Video and Speech Processing, Hong Kong.

[8] Makram Nabti and Bouridane (2007). An effective iris recognition system based on wavelet maxima and Gabor filter bank. IEEE trans. on iris recognition.

[9] Masek, L. (2003). Recognition of Human Iris Patterns for Biometric Identification.

[10] Narote et al. (2007). An iris recognition based on dual tree complex wavelet transform. IEEE trans. on iris recognition.

[11] Panganiban, A. (2009). CCD Camera with Near-Infrared Illumination for Iris Recognition System.

[12] Panganiban, A. (2010). Implementation of Wavelet Algorithm for Iris Recognition System.